University of Alberta

Retargeting of Virtual Reality Applications

by

Pablo Alejandro Figueroa ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Spring 2004

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-96265-2
Our file Notre référence
ISBN: 0-612-96265-2

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Current practices in development of Virtual Reality (VR) applications are very costly in general and difficult to adapt to various VR platforms. This thesis solves some of the issues related to the development of VR applications when a variety of hardware platforms is available, as in current VR labs and development sites. We make visible at a high level of abstraction the most important elements in the interface of a VR application. Current representation methods of VR are either too formal or too close to a programming language to be understood by users, which precludes the analysis, evaluation, and improvement of interface issues. We define a clean separation between different software components in a VR application and its associated semantics. This separation allows us to reuse VR components, without having to worry about unexpected side–effects. We also define a new way to transform an application from one hardware platform to another. We call this process retargeting, and it is based on our ability of component reuse and the high level of abstraction language we define. We separate two important roles in the development of VR applications. One is in charge of the overall architecture of the application. They pay attention to interface issues, requirements coverage, and component reuse. The second one is in charge of fine–detail development of components and its tuning to a particular deployment environment. We consider this separation an important way to handle complexity in the development process. It enables different people to concentrate on different issues and at the same time collaborate on the development.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To Blanca, Claudia, and Monica.

To all people that I remember, and to all people I do not, for their contributions to this project I call life.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

The field of Virtual Reality (VR) is now more than 30 years old, and in several ways is still in its infancy. After initial overrated publicity, motivated in part by spectacular applications envisioned by science fiction authors, and the frustration that followed when expectations were not met fast enough, VR is finally becoming a real technology with clear solutions in some specific industries. VR applications in car prototyping, oil exploration, military simulators, entertainment, and fear treatment, among others, have demonstrated that VR technology has distinct advantages in certain fields, and that more research is required to broaden its potential use.

One problem with VR technology is that application development is still a daunting task, partially due to the lack of mature and widely accepted development tools and methodologies. In order to achieve quality in a VR application, several novel input and output devices should be seamlessly integrated; as well, users should be able to interact with information in powerful ways. Such a deployment is complex due to hardware calibration issues, software integration of frameworks and libraries, and the inherently technical and interwoven nature of VR technology. Current practices are very costly in general and difficult to adapt to various VR platforms.

This thesis solves some of the issues related to the development of VR applications when a variety of hardware platforms is available, as in current VR labs and development sites. The main contributions of this thesis are:

- We make visible at a high level of abstraction the most important elements in the interface of a VR application. Current representation methods of VR are either too formal or too close to a programming language to be understood by users, which precludes the analysis, evaluation, and improvement of interface issues.

- We define a clean separation between different software components in a VR application and its associated semantics. This separation allows us to reuse VR components, without having to worry about unexpected side–effects.

- We define a new way to transform an application from one hardware platform to another. We call this process retargeting, and it is based on our ability of component reuse and the high level of abstraction language we define.

- We separate two important roles in the development of VR applications. One is in charge of the overall architecture of the application. They pay attention to interface issues, requirements coverage, and component reuse. The second one is in charge of fine–detail development of components and its tuning to a particular deployment environment. We consider this separation an important way to handle complexity in the development process. It enables different people to concentrate on different issues and at the same time collaborate on the development.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The ideas presented in this thesis evolved from a thorough study of previous results in the area of development environments for VR applications, our development of four environments with key ideas, and our development of an application as a proof of concept for retargeting. Our first environment used C++ and Python as programming languages, which allowed us to change the behavior of the application at runtime. The second environment used C++ and MRToolkit, a library developed here at the University of Alberta, and went further into the complexity of a program with replaceable behavior and defined the main concepts and structure in our representation of VR applications. We took these previous experiences and added the important concept of separation of roles for development in our third development environment, based on the Java programming language and XML. Finally, a fourth development environment based on C++, XML, and Performer is under construction, which provides developers of our CAVE–like environments with a platform that supports the concepts in this thesis. We made some attempts to use other libraries and domains, such as information visualization based on the VTK library, but we decided to concentrate most of our efforts on scene–graph–based technologies such as Java3D and Performer as a way to limit the scope of this thesis. After the second prototype, we also started a process of refinement of our domain analysis by describing it in a formal language (Z), and developing architectural reviews with peers based on such a description. More than 80 person–hours were spent in this review process.

The following sections give further explanation of our main contributions, which will be treated in full detail in the rest of this document.

HCI Elements of VR Applications

In order to emphasize the main concepts in a VR application from the point of view of its interface, we have developed a domain–specific language in which the domain is the set of virtual reality applications. Domain–specific languages allow developers to describe solutions to problems in a language closer to the domain than to a generic programming language. Current programming environments for VR are limited in domain–specific concepts: some of them cover a partial set of concepts or their implementation as a framework on top of a programming language. This usually requires high level programming skills.

Current VR specific languages usually describe in great detail the geometry of the solution and sometimes the behavior expected from the application. However, they usually limit the set of input and output devices to a few, or the capabilities of available hardware to the ones that are standard to several installations. Our novel approach describes the architecture of an application in terms of abstractions of all important elements one can find in a VR application interface: devices, behavior, and content. Such a novel language gives developers the possibility of designing solutions for any VR installation that might be available without going into excessive amount of details.

We have developed a formal description of our language in Z, a formal programming language with a clear semantics. This description allows us to understand concepts related to the state and dynamic behavior of a system without referring to subtle details in any particular implementation. This language also allows developers to implement such concepts in any VR platform available. Nevertheless, our two last implementations in Java3D and in C++ fulfill a subset of the concepts in the Z language description and can be taken as a functional reference.

Software Components for VR Applications

Component–based development allows the construction of applications based on reusable components. A component should clearly state its interface [1], its state information, and

[1]The interface of a component is the set of characteristics that define its input and output.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

possible side effects. Most of the current toolkits for VR development organize applications in some modular way, based on particular programming language constructs, but these modules usually have internal side-effects and rely on a fixed, non-scalable execution model, which makes them very difficult to reuse.

We introduce a modular way to construct VR applications based on our domain specific language and its well defined components. Components for devices, behavior, and geometry can be defined and used, at a higher level than a programming language such as C++ or Java. Such components can be much more relevant to developers, since they relate directly to concepts found in the VR domain, thus applications can be structured in a way closer to the domain. They also distance the developer from details of the execution model, so both sequential and concurrent models are allowed in the implementation and details are hidden inside components.

Retargeting of VR Applications

Retargeting is a term used in the compiler community that refers to the adaptation of a particular source code to the characteristics of a particular processor, therefore an implementation uses the most of an architecture. It relates to all methods for transforming an implementation to the most suitable structure in a target platform.

Our novel solution for VR application development makes possible the retargeting of VR software to the particular capabilities of a VR environment. Since a VR application is described in a high level language, it is possible to accommodate its implementation to the particular characteristics of an installation. In addition, since domain specific components like devices and behaviors are easily recognized, it is possible to transform an application from one installation to another, by replacing devices and behavior while keeping support for the same tasks. The idea of retargeting in the field of VR is novel, and it opens possibilities for VR development where several hardware configurations can be tested and compared.

Roles in VR Application Development

One way to reduce complexity of a problem is to divide it into distinct subproblems, each one solvable by people with certain kinds of skills. This simple idea can be used to control the complexity of virtual reality applications, allowing developers with different skills to attack different issues in a development effort.

Our approach differs from previous attempts to solve this problem in the definition of different roles for developers, in such a way that responsibilities and skills can be differentiated. Previous attempts could separate concerns between development groups, but they assume similar skills in each group. In our approach we define the following roles, each one with different skills: designers, developers, and technicians. This reflects current practice in teams building VR applications.

A designer of a VR application concentrates on the application domain and is a user of a domain specific language that gives her access to the technology. Designers know what VR applications can do, their limitations, and possibilities, but do not necessarily have programming skills to code a system using languages such as C++ or Java. A designer is also able to analyze the application domain, and translate it into a solution based on interconnected components in the VR domain. Designers are capable of defining requirements for new components, or reuse previously defined ones, without a deep understanding of how they work. They know about a VR specific language to express VR applications, and they use such a language to express requirements to developers. A designer is more interested in the overall behavior of the application and in its usability characteristics, rather than in the nuts and bolts of the implementation. Their knowledge is platform independent, but they know the particular characteristics of each VR installation and how to take advantage of such differences in an application.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A developer is the role assigned to people who know how to program VR applications, who know about the technology involved, and how to use it. Developers are proficient in programming languages such as C++ or Java and are able to implement functionality required by designers. Developers are interested in performance and frame rate, and they are capable of tuning an implementation in order to get the most out of a VR installation.

A technician in a VR installation is responsible for calibrating sensor and device performance. They know about low level drivers, devices, equipment, and networking. They are responsible for hardware tuning and general setup for VR applications in an installation.

In this thesis, we concentrate on the separation of concerns between designers and developers, since this division is the most difficult to address. The work of technicians is considered highly specialized, and it is closer in skills to what developers do. For this reason both will be considered as one role.

Organization of This Document

Chapter 2 describes previous work that is related to our thesis. Several points of view are analyzed, one per each of our main contributions. Chapter 3 introduces our domain-specific language, in both an informal and formal way. Chapter 4 describes the development methodology that is related to our language, and an example on how our test application was developed. Chapter 5 describes in more detail some of the work around our domain-specific language, including two VR applications described in such a language, and a user study that was made possible due to our rapid retargeting process. Chapter 6 describes in detail the development environment we envision, the tools we have developed, and some guidelines on how to develop further the key tools in the future. Finally, we present our conclusions and new research directions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

Related Work

Our work improves the techniques, tools, and procedures involved in the development of VR applications. We describe in this section the current literature in VR development, as well as how software engineering techniques apply to VR projects. We first give an overview of the most important ideas in current toolkits for VR development, and describe examples of applications. We then analyze the most important results in the main areas of our contribution, i.e. domain specific languages, software architectures, execution models, retargeting, and development roles for VR. We then compare current VR toolkits that one can find in the literature. Along this comparison, we restate our motivations and our main contributions.

2.1 An Overview of VR Toolkits

There are many toolkits for VR development, with different scope and complexity. Some allow users to configure a wide spectrum of issues, while others hide some decisions from developers in order to reduce complexity. Some environments are tailored to a particular hardware platform, and others allow developers to use a wide range of input and output devices. This section describes the most important features of commercial products and research activities around VR in the last 15 years. At the end, comparison will be done in terms of the issues developers concentrate in, and how one can deal with this wide variety of platforms.

Initial efforts in this area were very ambitious and generic. AVIARY [86, 105] is an example of such a system, a software architecture for a distributed, generic, virtual reality system. AVIARY is a distributed system with time management capabilities, shared world management among its processes, and support for multiple virtual worlds. Devices are handled by input processes that are able to inform changes to other interested processes. Output processes handle output devices and a Virtual Environment Manager process handles the consistency of the application state. Not many details are available about implementation of applications using AVIARY, or about its extension or component reuse mechanisms.

A popular system at the beginning of the nineties was the MR–Toolkit [81], a C library for the development of VR applications. The system runs several processes, one for the application computations, one for device polling, and a slave process for output tasks. Developers needed to be proficient with configuration files for some devices. Developers could also extend the system by adding functions for new devices, following a certain set of conventions for registering callbacks at runtime. Listing 1 shows an example of a MR–Toolkit application, described in [81]. Two programs are represented in the same source code, a program in charge of reading devices and updating the simulation (master), and a program that updates its state after each computation (slave). Such a configuration is able to update two displays in a stereo setup. From the listing, one can notice the required operations for adding devices (lines 26,27), several configuration tasks (lines 29–37), and the main loop

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(lines 39–54), which reads devices, translates input signals into gestures, and updates the virtual representation of a hand. Extensions to this program require changes in the code, in order to accommodate new devices, new behaviors, or new objects to be displayed.

CAVELib [100] was the first library developed for CAVE–like environments, and it has been evolving since 1992. It helps developers isolate from issues related to the management of several displays, several input devices, and multiprocessing capabilities of the underlying platform. Listing 2 shows the same example described in [51], based on OpenGL [1].

The application reads configuration files (line 9), initializes information shared among all application processes (line 10), the application environment (lines 12,13), and the rendering function (line 14), and the simulation cycle (line 15). Once the initialization tasks are done, it will compute and show the next simulation cycle until the escape key is pressed (lines 17–19). Configuration files allow developers to activate hardware options, select device or network calibration options, and turn on or off debugging capabilities. Details about rendering on several displays, and synchronization of several processes are hidden from developers. Support for new devices or new interaction techniques is limited to some well-known commercial alternatives.

The Virtuality Builder II (VB2) [45] is an architecture that defines a process for each key VR task (device input, application computation, rendering), and a constraint–based mechanism to change propagation between entities in the application. The application code is organized in terms of active variables, daemons interested in variable changes, constraints, and reaction objects. Active variables and constraints can be encapsulated inside classes, and transactions can be defined in order to group several atomic operations over active variables in a class. Reaction objects can be called at the end of transactions for invariant checking. A global time is maintained, and incremented every time constraints are executed. VB2 appears to be the first system to use the concept of hierarchical constraints for relationships between elements in an application, and the concept of a *virtual tool*, a reusable widget that allows users to visually handle a certain type of information. VB2 is written in Eiffel, and uses a sophisticated algorithm for constraints resolution at every frame.

Performer [79] is a framework for manipulating scene graphs, created by SGI. It features an architecture capable of fixed frame rates and frame synchronization between several machines, suitable for high–end graphics applications. The application data is organized in a scene graph, a tree–like structure that organizes geometry and other media in the virtual space. Performer's execution model is a pipeline of three tasks: CULL, DRAW, and APP, that make geometric culling, scene drawing, and application computations, respectively. User specific functions can be added as callbacks in each one of these stages, or in user defined traversals of the entire scene graph. Listing 3 shows the general structure of a Performer application [2].

The initialization process, which may include the execution of several threads, is done by the framework. A scene is created (line 10), which is the root for a tree that represents the geometry in the world. Pipes and channels, the main abstractions for the hardware architecture, are created after (lines 13–16). At the end, the application enters the simulation loop, in which images are produced, drawn in a second buffer, and shown in the possibly several screens of the system (lines 16–21).

Avocado [96] is a framework for the development of distributed VR applications. Applications can be written in C++ or Scheme, a scripting language. It is implemented on top of Performer, and deals with the complexity of object distribution among several computers. Its architecture uses inheritance to deal with new types of distributed objects, and a callback mechanism from Performer for receiving changes from the world.

Lightning [13] features a dataflow–based execution model and a programming interface in both Tcl and C++. Listing 4 shows parts of an application in Tcl, mentioned in [13], that defines an input sensor for the head position and orientation, some frustum parameters, and

[1] CAVELib can also use Performer as rendering engine.

[2] Complete examples can be found in a Performer installation, under the directory /usr/share/Performer/src/pguide/libpf/C++/.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 1 A MR–Toolkit Application

```
#include <MR/mr.h>
#define machine ''tawayik''
#define program  ''hello_world''
extern Gtable gtables [];

main (argc, argv)
int argc ;
char *argv [ ];
{
int quit_id, count = 0;
Program slave;
Data shared_cnt;
Hand hand ;
Gtable gst_tbl ;
Gesture_event usr_gst;

  /* Configuration section */
  MR_init (argv [0] ) ;
#ifdef MASTER
  MR_set_role (MR_MASTER);
#else
  MR_set_role (MR_SLAVE) ;
#endif MASTER
  slave = MR_start_slave( machine, program) ;
  shared_cnt = MR_shared_data (&count, sizeof (count) , slave, MR_FROM) ;
  MR_add_device_set (EyePhone);
  MR_add_device_set (DataGlove);
  EyePhone_slave (slave) ;
  MR_configure ( ) ;

  /* Computation section */
  read_gesture_file( ''my.gst'' );
  assign_gesture_ids () ;
  gst_tbl = gtables[0];
  quit_id = get_gesture_id(''select'') ;
  set_room_reference(1.0, 1.5, 2.0);
  map_reference_to(0.1, 0.0, 0.5);

  while (1) {
    update_hand( );
    hand = get_hand();
    usr_gst = recognize_gesture(gst_tbl) ;
    if (MR_get_role() == MR_MASTER)
      if (usr_gst->id == quit_id)
        MR_terminate(0);
    MR_start_display() ;
    count++ ;
    if (MR_get_role() == MR_MASTER)
      send_shared_data(shared_cnt );
    else
      shared_data_sync(shared_cnt );
    draw_hand(hand) ;
    MR_end_display ();
  }
}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 2 A CAVELib Application

```
#include <cave.h>
void app_shared_init();
void app_compute_init();
void app_init_gl();
void app_draw();
void app_compute();

main(int argc,char **argv) {
        CAVEConfigure(&argc,argv,NULL);
        app_shared_init(argc,argv);

        CAVEInit();
        CAVEInitApplication(app_init_gl,0);
        CAVEDisplay(app_draw,0);
        app_compute_init(argc,argv);

        while (!getbutton(ESCKEY)) {
                app_compute();
        }

        CAVEExit();
}
```

connections between the head sensor and the frustum parameters.

The structure of the application looks easier to understand than in previous systems developed in C or C++, since the scripting language shows just the high level elements and their connections, and hides the intrinsic complexity of device drivers and application code. Events from devices are propagated through the routes (lines starting with `ltroute`), in a way that guarantees that previous nodes are executed first. Devices are read in different processes, so different update rates are supported.

WorldToolkit (WTK) [78] is a C/C++ based library for the development of portable VR applications. It provides fairly sophisticated visual effects and support a wide variety of hardware and file formats (for geometry description). Listing 5 shows an example of a program using WTK [77]. An application in WTK defines the file for the objects in the world (line 19), devices to use (line 20), and tasks associated to objects (lines 22, 27–30). The framework hides the complexity involved in the environment setup (line 17) and simulation execution (line 25).

MASSIVE-2 is a framework for multi-user, VR applications. It concentrates on the "... *mechanics* of managing awareness and communication among *human participants* in crowded [Collaborative Virtual Environments]" [11, p. 59]. It introduces important concepts for the management of multiple users in a VR environment, such as the concept of aura — the volume of interest of an object —, and the concept of third–party objects — mediators that represent groups of people. Applications are written in C or C++, and the mechanisms for third–party objects are transparent to the programmer.

Alice [97] is a 3D programming environment with a simple user interface that allows people with no programming experience to create their own programs. The development process is the following [30]: Users first define the scene of their world, by selecting objects from a rich library, included in the environment. After the scene is defined, users can add behavior by creating scripts in an adapted version of Python. Listing 6 shows an example of a small script in Alice, that makes a bunny beat a drum [30, p. 489].

A program in Alice is very intuitive, since its language hides the complexities of 3D

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 3 A Performer Application in C++

```
int main (int argc, char *argv[])
{
    // Initialize and configure OpenGL Performer
    pfInit();
    pfMultiprocess( PFMP_DEFAULT );
    pfConfig();
    pfFilePath(".:/usr/share/Performer/data:../../../../data");

    // Create a scene, configure it and add some geometry and lights
    pfScene *scene = new pfScene;
    // ...

    pfPipe *pipe = pfGetPipe(0);
    pfChannel *chan = new pfChannel(pipe);
    chan->setFOV(60.0f, 0.0f);
    chan->setScene(scene);

    while (!exitFlag)
    {
        pfSync();
        pfFrame();
        UpdateView();
    }

    // Terminate parallel processes and exit.
    pfExit();
    return 0;
}
```

Listing 4 A Lightning Application in Tcl

```
# definition of the trackinginput
lttrackersensor headsensor -device bird -sensor 1 ...
...
# definition of the cameras (origin: cave center)
ltcamera frontcamera -position  ''0 150 0''  -orientation  ''0 0 0''
ltcamera leftcamera -position  ''150 150 0''  -orientation  ''-90 0 0''
ltcamera rightcamera -position  ''-150 150 0''  -orientation  ''90 0 0''
ltcamera bottomcamera -position  ''0 0 -150''  -orientation  ''0 -90 0''

# connect cameras with head tracker
ltroute  ''headsensor position''   ''frontcamera position''
ltroute  ''headsensor orientation''   ''frontcamera orientation''
ltroute  ''headsensor position''   ''leftcamera position''
ltroute  ''headsensor orientation''   ''leftcamera orientation''
ltroute  ''headsensor position''   ''rightcamera position''
ltroute  ''headsensor orientation''   ''rightcamera orientation''
ltroute  ''headsensor position''   ''bottomcamera position''
ltroute  ''headsensor orientation''   ''bottomcamera orientation''
...
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 5 A WTK Application

```
/* SAMPLE PROGRAM */

/*
* simple.c
* Usage: Use the mouse buttons to fly around a spinning planet.
*/
#include "wt.h"

void spin(WTnode *);
#define Y_AXIS 1
void main(int argc, char *argv[])
{
  WTnode *root;
  WTnode *planet;
  WTsensor *sensor;  /* the Mouse */
  WTviewpoint *view; /* the Viewpoint */
  WTuniverse_new(WTDISPLAY_DEFAULT, WTWINDOW_DEFAULT);
  root = WTuniverse_getrootnodes();
  planet = WTmovnode_load(root, "PLANET.NFF", 1.0);
  sensor = WTmouse_new();
  view = WTuniverse_getviewpoints();
  WTviewpoint_addsensor(view, sensor);
  WTtask_new(planet, spin, 1.0);
  WTuniverse_ready();
  WTuniverse_go(); /* Starts simulation */
  WTuniverse_delete(); /* All done */
}
void spin(WTnode *planet)
{
  WTmovnode_rotateaxis(planet, Y_AXIS, 0.01);
}
```

Listing 6 An Alice Application

```
ArmsOut = DoTogether(
        Bunny.Body.LeftArm.Turn(Left, 1/18),
        Bunny.Body.RightArm.Turn(Right, 1/8) )
ArmsIn = DoTogether(
       Bunny.Body.LeftArm.Turn(Right, 1/8),
       Bunny.Body.RightArm.Turn(Left, 1/8) )
BangTheDrumSlowly = DoInOrder(
                  ArmsOut,
                  Armsln,
                  Bunny.PlaySound('bang') )
BangTheDrumSlowly.Loop()
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

programming. An interesting feature is that users do not require knowledge of the object's absolute coordinates, since all animations can be described in terms of object–centric directions (i.e. forward and backward), or relative to other objects in the environment.

VRJuggler [12] is an open source framework for the development of immersive environments. VRJuggler provides several features, such as operating system independence, device abstraction, support for several graphic APIs, and hardware architecture abstraction. VR applications in VRJuggler are written in C++, by creating subclasses of key elements in the framework. Excerpts of `cubes` [33], an example in the VR Juggler distribution, is shown in listings 7 and 8.

Listing 7 Main function in a VR Juggler Application

```
 1  int main(int argc, char* argv[])
    {
       vjProjection::setNearFar(0.01, 10000.0f);

 5     vjKernel* kernel = vjKernel::instance();
       cubesApp* application = new cubesApp(kernel);

       for(int i=1;i<argc;i++)
          kernel->loadConfigFile(argv[i]);
10
       kernel->start();
       kernel->setApplication(application);

       while(1)
15     {
           usleep (250000);
       }
    }
```

The main function in a VR application defines parameters for the display (line 3), initializes the VR Juggler kernel and the application (lines 5, 6, 11, 12), loads configuration files with details about devices (lines 8,9), and runs the application in a different thread than the one for the main function. Most of the application–specific code goes inside the overridden functions of the application subclass (cubesApp, in this case), which is shown in Listing 8.

`UserData` holds the navigation techniques and devices for a particular user (lines 1-13). A subclass of `vjGlApp` [3] redefines common functions with application specific code: `init` for device initialization, `preFrame` for computations after reading devices, `draw` for scene drawing, `intraFrame` for operations while the scene is drawn, and `postFrame` for operations after the scene is drawn. This class also holds data structures for the output devices in use and the registered users. Several details about devices are hidden in configuration files, and other details about devices, interaction techniques, and users are defined in C++ code.

X3D [104], a rewrite of VRML with XML–based syntax, provides a common ground for description of 3D objects and simple animations, suitable for browsing over the Internet. Most of its definitions are related to geometry, but there are also ways to describe sound and scripts of animation. Listing [37] shows an example in X3D that shows the letters VRML on the screen.

The navigation technique is defined in line 3, from a predefined set of constants. The code then defines how to position each letter in the screen [4]. Absolute 3D coordinates are

[3]`vjGlApp` is a common placeholder for all OpenGL–based applications. There are others for Performer–based and OpenSG–based applications.

[4]Geometry for letters is loaded from the mentioned files.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 8 An application subclass in VR Juggler

```
class UserData
{
public:
   UserData(vjUser* user, std::string wandName, std::string incButton,
            std::string decButton, std::string stopButton);
   void updateNavigation();
public:
   // Devices to use for the given user
   vjPosInterface       mWand;                  // the Wand
   vjDigitalInterface   mIncVelocityButton;     // Button for velocity
   vjDigitalInterface   mDecVelocityButton;
   vjDigitalInterface   mStopButton;            // Button to stop
};

class cubesApp : public vjGlApp
{
public:
   virtual void init();
   virtual void preFrame();
   virtual void draw();
   virtual void intraFrame();
   virtual void postFrame();

   vjGlContextData<ContextData>  mDlData;       // Data for display lists
   vjGlContextData<ContextData>  mDlDebugData;  // Debugging display lists
   std::vector<UserData*>        mUserData;     // All the users in the program
};
```

Listing 9 An Application in X3D

```
<X3D>
  <Scene>
    <NavigationInfo headlight="false" type="EXAMINE ANY"/>
    <Group>
      <DirectionalLight DEF="SceneLight" ambientIntensity=".7"
        direction="1 -1 -1" intensity="1"/>
      <Transform DEF="VRML:" translation="-5 0 0">
        <TouchSensor DEF="vrmlTouch"/>
        <Transform DEF="char_V" translation="1.416 -.291 0">
          <Inline url="letter_V.wrl"/>
        </Transform>
        <Transform DEF="char_R" translation="1.946 -.048 0">
          <Inline url="letter_R.wrl"/>
        </Transform>
        <Transform translation="2.621 -.048 0">
          <Inline url="letter_m.wrl"/>
        </Transform>
        <Transform translation="4.042 -.048 0">
          <Inline url="letter_L.wrl"/>
        </Transform>      </Transform>    </Group>    </Scene>
</X3D>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

required for positioning objects, and DEF statements allow cross references in other parts of the world, or scripts.

CONTIGRA [35] is a XML application on top of X3D that give users components for interaction techniques and control widgets. Behavior3D [36] is a library with a set of reusable components for behavior. Component classes and instances are written in XML documents, which follow CONTIGRA schemas. Connections between geometry and behavior are written as xpointers [28], the standard way to refer to parts of XML documents. Listing 10 shows excerpts of an application in CONTIGRA [36].

Two X3D touch sensors trigger the animation of a laptop [5]. A state machine node in Behavior3D defines the order for the animations (lines 4–8), by defining which sensor triggers which animation. Lines 10–14 define the animations that open and close the laptop. Finally, lines 16–27 define animations to open and close the keyboard (removable in this example). The core concepts in Behavior3D are animations, sequences of actions, and state machines. On top of them, other behaviors can be defined. Behavior3D targets 3D applications running over the web, in standard PCs, due the intrinsic characteristics of X3D and its interaction model.

Massink, Duke, and Smith [61] make a proposal for structuring and formalizing the interaction techniques behavior. Their formal proposal is based on the concept of *hybrid models*, a model that contains continuous and discrete components at the same time. The notation they chose for the representation of interaction techniques is HyNet, a hybrid extension of Petri Nets. Developers can completely specify interaction techniques in HyNet, analyze them, and compare them in a platform–independent manner. While this formalism has some interesting characteristics for the specification of interaction techniques, Petri Nets are too detailed for designers needs.

Jacob, Deligiannidis, and Morrison [53] present the PMIW management system that describes an application with a combination of two notations: a dataflow that describes the continuous flow of information from devices, and a state machine that describes different modes in the execution. Transitions in the state machine are triggered by events from the interface. Execution of each node in the dataflow depends on states in the state machine, so it is possible to shut down parts of the dataflow by changing states. This notation provides enough detail for representing an interaction technique, but it might be too detailed for designers who want to use it without knowing its internal mechanism. Further research is also required to test this notation in complex VR applications.

The Virtual-Reality Peripheral Network (VRPN) [75] is a C++ framework that handles devices by the definition of virtual devices — abstractions for the main types of information that devices can provide —, connected to real devices by a name and an index (the device's sensor number). Developers can add new devices as composition of previous ones, create new classes of devices, and receive events in specially defined callbacks. An example of how such devices are used is shown in listing 11 [75, p. 59]. An application in VRPN is a set of callbacks that handle events from one or more virtual devices. These callbacks are registered in the system (line 10) and the system will redirect the events to such a function. Events can be time–stamped, so it is possible in theory to correlate input from several devices.

Other ideas have influenced our results in our proposal. Systems like JADE [67] and Bamboo [103] are systems that can be reconfigurable at runtime by adding or removing modules. Our system also defines modules for the most important elements in a VR application, but with a more complex interface, related to the application execution, not only to the dynamic loading behavior. Other component technologies in programming languages such as Java have been used for 3D applications [41], but they are specific to these environments, and limited by the inherent performance characteristics of the generic component approach. The requirement of event models more complex than the one in Windows–Icons–Menus–Pointer (WIMP) applications have been described before, and some solutions have been described, such as the one in [54] that manages detection of composed events at the

[5] Connections to the laptop geometry are not shown here.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 10 A CONTIGRA Application

```
<TouchSensor DEF="LCD_Sensor"/>
<TouchSensor DEF="Keyboard_Sensor"/>

<StateMachine stateCount="3" transitions="
    1 2  LCD_Sensor.touchTime  OpenLaptop.startTime,
    2 1  LCD_Sensor.touchTime  CloseLaptop.startTime,
    2 3  Keyboard_Sensor.touchTime  OpenKeyboard.startTime,
    3 2  Keyboard_Sensor.touchTime  CloseKeyboard.startTime"/>

<AnimateRotation key="0 1" to="1 0 0 0, 1 0 0 -1.7"
    cycleInterval="2" DEF="Openlaptop "/>
<AnimateRotation key="0 1" to="1 0 0 -1.7, 1 0 0 0"
    cycleInterval="2" DEF="CloseLaptop"/>

<Sequential DEF="OpenKeyboard">
    <AnimateTranslation key="0 1" to="0 0 0, 0 0.05 0"
        cycleInterval="1" />
    <AnimateRotation key="0 1" to="1 0 0 0, 1 0 0 -1.5"
        cycleInterval="1" />
</Sequential>
<Sequential DEF="CloseKeyboard">
    <AnimateRotation key="0 1" to="1 0 0 -1.5, 1 0 0 0"
        cycleInterval="1" />
    <AnimateTranslation key="0 1" to="0 0.05 0, 0 0 0"
        cycleInterval="1" />
</Sequential>
```

Listing 11 A VRPN Application

```
#include  vrpn_Tracker.h

void handle_pos(void *, const vrpn_TRACKERCB t) {
    printf(``Pos, sensor %d = %5.3f, %5.3f, %5.3f\n'',
             t.sensor, t.pos[0], t.pos[1], t.pos[2]);
}

main() {
    vrpn_Tracker_Remote *tkr = new vrpn_Tracker_Remote( ``Tracker0@myhost'' );
    tkr->register_change_handler(NULL, handle_pos);
    while (1) { tkr->mainloop(); }
}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

operating system level, while the application concentrates on behavior. We consider such a solution partial, since it does not consider relationships and event–based communication between behavioral components. Some preliminary ideas of classes, encapsulation, and component linking applied to 3D applications appeared in [90]. The concept of dataflow have appeared in visual programming environments for VR such as [89], or in programming environments such as VRML, but with limited semantics for modules in the dataflow.

It is useful to compare how much information is required in order to produce a program in each VR toolkit above. Table 2.1 shows this comparison, for the VR toolkits with listings in the previous overview. We have added a line for InTml, the system that will be described in this thesis. In summary, one can conclude that most environments with wide coverage of hardware platforms require developers to take decisions on many detailed aspects, and to know a rather complex programming language. Environments with limited hardware coverage tend to ease the developer's work, by hiding many issues. InTml provides a high level solution that both eases the work of developers and allows them to use any hardware platform.

Toolkit	LE	HC	IS	MP	DR	DD	ME	DE	DW	SY	ST	CC	GD	NT
MRToolkit	░	▩	■	■	■	■	■	■	■	■	■			
CAVELib	░	■	■		■		■				■			
Performer	░	░	■	■	■		■		■	■	▩		■	
Lightning	▩	░			■	■					▩	■		
WTK	░	■	■		■		■				▩		■	
Alice	▩	░									▩		░	
VR Juggler	░	■	■		■		■		■		▩			
X3D	■	░									▩	■	■	■
VRPN	░	░	■		■			■			▩	■	■	■
InTml	■	■			■						■	■		

Table 2.1: Coverage of Interface Elements in VR Toolkits. Short names in the first row are as follows: programming language ease of use (LE), hardware coverage (HC), initialization and setup (IS), multiprocessing (MP), device registration (DR), detailed device configuration (DD), main execution cycle (ME), device–related event handling (DE), drawing (DW), synchronization management (SY), simulation task modeling (ST), connections between components (CC), geometry details (GD), and navigation techniques (NT). A darker gray means a better option than a lighter one, from the viewpoint of a VR developer.

2.2 Domain Specific Languages

Concepts in a Domain Specific Language (DSL) are specialized to a particular field, and help designers to concentrate on the solution space rather than both the solution space and the intricacies of a programming language. A solution for a problem can be written in a more succinct and clear way in a DSL than in a general purpose language. It is important that a DSL can be extendible, so users can add their own concepts to the domain. Modularity is also important, so different levels of specialization can be defined in the same domain [34, p. 138]. An example of modularity and different levels of abstraction is the `SimpleUniverse` class in Java3D, that facilitates the use of Java3D concepts related to display management in the case of a standard PC platform. If the application is targeted to the standard PC platform, the implementation can use `SimpleUniverse` instead of the more complicated structure in terms of lower level classes.

From the previous analysis of VR toolkits, we notice that traditional languages for VR applications are C++ and Java, enriched with special–purpose frameworks. There are few development environments that use other languages, such as Python [97], Tcl [39], Modula-3 [59], or Scheme [5]. A developer working on one of these environments should be proficient

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in the language they use and in the library or framework that gives access to VR concepts. Extensions to the DSL are defined in terms of classes or functions in the general–purpose language. Usually, these languages are not modular, so a developer sees just one level of concepts.

In contrast, VRML [21] is a DSL suitable for Web–based VR applications that does not require extra knowledge in programming languages. Its concepts are mainly related to the geometrical characteristics of the virtual objects to be shown, and extra behavior can be defined in JavaScript or Java. Developers who want simple animated models in the Internet can use it without the extensions, so programming skills are not required. Also, it is assumed that the learning curve for VRML is easier than the ones for traditional languages in VR. However, modeling has proved to be easier with the help of tools, such as Maya [3], 3D Max [2], or Blender [14]. Hence, it is more efficient to learn to use one of these editors than to write raw VRML. Extensions of VRML are created with the `PROTO` statement, so developers can enrich the language using the same language. VRML is not modular, but the new version of VRML, X3D [104], defines several layers of functionality in what they call profiles. In this way, developers interested in a subset of the functionality can avoid the concepts in other profiles.

In this thesis, we present a new DSL for the general architecture of a VR application, where details are not as important as the overall relationships between devices, behavior, and content are. Our approach gives developers a high level language in which complete VR applications can be described, without going into details about hardware calibration and setup, behavior algorithms, or special content details. We allow developers to concentrate on the definition of the main elements in a VR application and their relationships. The details are solved in the lower layers of the infrastructure. For example, our developers can use novel devices without worrying about the details of the connection, in a similar way to VRML developers who can create touch–sensitive geometry without knowing the particular algorithms for collision detection. We think our DSL allows non–programmers to design entire VR applications, and collaborate with programmers in such a way that a design can implement these functionality in an optimal way.

Our DSL, the Interaction Techniques Markup Language (InTml), is defined using an XML language with a defined schema. This allows basic type checking using standard XML tools. InTml allows developers to define new devices, new interaction techniques, and new content, so that the language can be enriched. Since InTml applications can be defined in terms of elements in a library of reusable components, it is possible for a developer to create an entire application without knowledge of traditional programming languages, or specific details about the implementation. When new elements are desirable, a development process defines how such elements can be implemented as a collaboration between designers and coders [6]. Table 2.2 shows how other development environments cover these interface elements, and which programming language they use. In general, we think InTml provides a higher level language with a better coverage of devices, behavior, and content references.

[6]The development process is defined in Section 4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Toolkit	Lang.	Devices	Behavior	Content
MRToolkit	C	▓	░	░
CAVELib	C	▓	░	░
VB2	Eiffel	░	▓	░
Performer	C++	░	░	■
Avocado	C++,	░	░	░
Lightning	C++,	▓	▓	░
WTK	C++	■	░	■
MASSIVE-2	C++	▓	░	░
Alice	Phyton	░	■	▓
VRJuggler	C++	■	░	░
X3D	XML	░	▓	■
CONTIGRA	XML	░	■	■
HyNet	Petri Nets	░	■	░
PMIW	Dataflows and State Machines	░	■	░
VRPN	C++	■	░	░
InTml	XML	■	■	▓

Table 2.2: Coverage of Interface Elements in VR Toolkits. A darker gray means a better option than a lighter one, from the viewpoint of a VR developer.

Currently, InTml is defined as just one module. However, its XML–based implementation will allow the creation of new modules in the same way that X3D does.

2.3 Software Architectures for VR Applications

Bass, Clements, and Kazman [9, p.23] define a software architecture as follows:

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

A software architecture helps developers, users, and other people interested in a software product to understand its high–level design, the context of each component, and how development work is divided among peers. It also makes clear the major system properties [29] and helps in the reuse of components from systems with similar requirements [52].

Some general–purpose architectures, such as MVC [20, p. 125] and PAC [20, p. 145], have been proposed for the broader field of user interfaces, in which VR is an specialization. Although the problems they solve are also very important in VR, requirements of VR systems such as high throughput and simultaneous use of non–conventional devices are not addressed in these proposals.

Table 2.3 shows an overview from the software architecture viewpoint of toolkits described in section 2.1. The table shows the main emphasis in each toolkit, which components can be created or used, and what are the relationships between defined components.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Toolkit	Lang.	Domain	Component Types	Relationships
MRToolkit	C	Devices, 3D Displays	Devices Displays	Control flow Conf. files
CAVELib	C	Devices, 3D Displays	Devices Displays	Control flow Conf. files
VB2	Eiffel	Int. Techniques, 3D Widgets	3D Widgets	Constraints
Performer	C++	3D Graphics	Content, Callbacks	Scene links Callbacks
Avocado	C++, Scheme	Distribution	*From Performer*	
Lightning	C++, Tcl	Event Propagation	Devices	Routes
WTK	C, C++	Geometry Devices	Content, Simple Animations	Callbacks Control flow
MASSIVE-2	C, C++	Multi–user	Content	Callbacks
Alice	Phyton	Novel user programming	Behavioral Scripts	Callbacks
VRJuggler	C++	Devices, Multiplatform, Run–time Control	Devices, Rendering Syst.	Control flow, Callbacks Conf. files
X3D	XML Java	3D Graphics Behavior	Content Scripts	Routes Callbacks
CONTIGRA	XML	Control Widgets Behavior	Behavior Nodes *From X3D*	Routes Composition
HyNet	Petri Nets	Interaction Techniques	States	Discrete and Continuous Transitions
PMIW	Dataflows and State Machines	Interaction Techniques	States, Variables	Conditions, Links Transitions
VRPN	C++	Devices	Devices	Callbacks
InTml	XML	Devices, Behavior Content refs.	Devices, Behavior Content refs.	Routes, Composition

Table 2.3: Extension Mechanisms in VR Toolkits

Most of these environments concentrate on devices and content for VR applications. They present a rather low level interface for development, based on a programming language such as C++ or Java, or in formal languages such as Petri Nets and state machines.

An architecture for VR applications should present at the top level of abstraction all important concepts, not only content and devices, but also object behavior and interaction techniques. An architectural language for VR applications should represent such components and their relationships, without going into too much detail, or into the detailed design of a particular component.

InTml is an architectural language for VR applications. It allows the description of content, devices, and behavior as typed components. A VR application is defined as a set of interconnected instances of component types. InTml extends the Pipes and Filters architecture [20, pp. 53–70] for the composition of components in an application, and a type hierarchy for the design and reuse of types of components. Detailed design for components is defined outside of the language, and it is possible to count with different design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

methodologies for each type of component [7].

2.4 Execution Models

The execution model of a programming paradigm defines how constructs on such a paradigm are executed by a computer, and how such an execution will affect the state of a program. It defines the semantics of a document that represents a program.

As we have seen in section 2.1, there are several toolkits that provide support for VR development, with different scopes and execution models. Most current VR environments implicitly follow the execution model of traditional windows–based toolkits, as shown in Figure 2.1. Simply, a windows application receives events from a dispatcher that selects interesting events from a system queue. Usually, there is a fixed set of events which a window receives, from a fixed set of input devices. Some toolkits allow the creation of extensions for new devices or new events, but these capabilities target senior developers and they are rarely used.

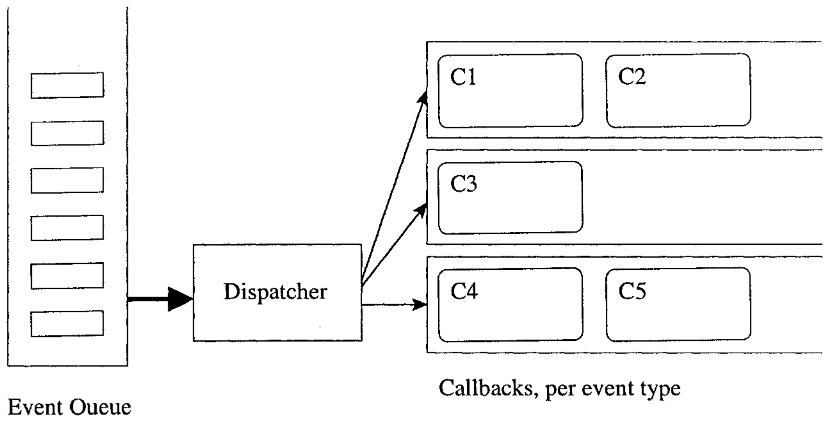


Figure 2.1: Callback based Architecture

Application code is written in special callback functions, that are registered in the system in some standard ways. Despite its success in standard interfaces, this architecture has the following limitations for VR applications:

- Addition of new events from novel input devices is a difficult task, so it is usually avoided by reusing events from standard devices that are not presently in use. This creates problems due to usability differences between devices, and conflicts if new and old devices want to be used.
- There are no provisions for more complex structures between callbacks. All callbacks are just at one level from the dispatcher, without structure between them.
- Interrelationships between callbacks are difficult to model, and are usually based on global memory, which is not a good solution from the software engineering viewpoint.
- There are limited possibilities for composition and reuse of third–party components due to the lack of scalability in the architecture. It is difficult to compose callbacks that were previously developed in isolation.
- Since all events are queued and serialized, there is no provision for treatment of simultaneous events from different devices with different generation rates.

[7]For example, content can be created with a special–purpose tool, while interaction techniques can be written in a programming language.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Our proposal adapts traditional execution models for Pipes and Filters architectures, such as Synchronous Data Flows [10], to the following characteristics of a VR application:

- Not all information from input devices should be processed. Depending on the speed of computations and the refresh rate of output devices, some information from input devices could be irrelevant or out of date. We allow filters to define an interval of time where all received information is considered simultaneous, so redundant information inside the interval can be eliminated. This information does not affect successive intervals, so discarded information do not affect future executions.
- Complex VR applications require dependencies among different tasks and interaction techniques. The callback model is difficult to scale to more complex structures, where dependencies among callbacks are required. A model based on Pipes and Filters can define better relationships between different behavior components in the system, and it exposes the coupling clearly.
- New input and output devices are common in new applications. It should be simple to add new devices to an application. Moreover, simultaneous events from different devices should be easy to detect. Filters with several input and output ports are our solution to this problem. They can model any type of device in an uniform way, and it is easy to create new types of filters for new types of devices. On the other hand, a filter interested in simultaneous events from different devices just needs to include them as input, and read all events received in a time interval, from all its inputs.

Some intrinsic characteristics of VR applications are still not directly addressed by the proposal presented here, such as the desirable fixed refresh rate for output devices. However, it is straightforward to integrate previous solutions to this problem to our proposal, such as the one presented by Shaw and Green [81], which decouples device reading from simulation execution.

A pipes and filters architecture also allows us to consider dynamic and static scheduling algorithms, in the case of a machine with several CPUs. This approach can not be implemented in current dataflow–based solutions such as VRML and X3D, due to intrinsic limitations on the order of execution of components in a program. Kwok and Ahmad [56] discuss several algorithms for static scheduling, and solutions for arbitrary graph structures with arbitrary computational costs per node such as CP/MISF and DF/IHS are promising for high performance solutions in VR.

2.5 Retargeting

Retargeting is the name used in the field of compiler design for techniques that generate code for different hardware architectures, without loosing performance or special features of each platform [8]. The concept of retargeting is similar to the concept of porting, most commonly used in the software engineering community. However, traditional ways to port systems do not accommodate to the particular features of an installation. For example, object oriented frameworks allow developers to port applications over a variety of platforms, where the framework has been implemented. However, a framework usually encapsulates the common features of all its implementations, so developers can not use special features of particular platforms. The performance of a program that uses the framework is usually lower than an application specially tailored for such environments.

There have been several approaches to compiler retargeting [24]:

- A common language can be created and interpreted at runtime for all implementations. The runtime environment is capable of translating this common language code to machine code in an efficient manner.

[8]It has also been used in computer graphics animation for the task of adapting a motion scheme of a character to different models. We will use a definition of retargeting closer to the one in compilers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- A compiler can generate source code suitable for several targets. Such code is then compiled by a compiler in the target machine. It requires the compiler to know details about target platforms, so the generation process is specially tailored.
- As a variant of the last method, a compiler can both generate code for a particular target platform and also derive the specific features of this target.

Since VR applications are still very diverse, and interaction techniques are not standardized for 3D applications, we decided to take a basic approach to retargeting, which can be automated later on. Since the variations from one platform to another are unknown, the retargeting process takes place at design time, re–shaping an InTml–based application from one platform, to the particular features of the new environment. Several compilers are implemented, one per VR platform, in order to specially tailor executable programs from InTml descriptions. We define a formal model in the Z language [87] with the overall semantics of an InTml program. In this way, it is possible to have a semantic reference apart from any implementation. We selected the Z language since it has a mature user community and it has been used in the past for the communication and understanding of static and dynamic properties of systems Our description is based on previous work by Philipps and Rumpe [68], and Lee and Parks [57]. Philipps and Rumpe define a method of specification refinement of Pipe–and–Filter architectures, from which most of our concepts are taken. There are some differences between these works: our purpose is not refinement but communication of concepts, we define an execution model more suitable to the concepts of VR we define, and concepts like objects, and object holders are novel. Lee and Parks define in great detail the concept of dataflow networks, and its semantics. Although our description is more similar to the one from Philipps and Rumpe, we borrowed the important concept of delays from Lee and Parks.

There have been some attempts to define a concept like retargeting in the computer graphics area. Scalable graphics is a field that studies methods for parallel rendering of scenes [9]. Some results have been achieved in this area [49, 42, 66, 64] which basically propose algorithms for load balancing the rendering task over several computers. Application retargeting in VR requires this type of rendering solution, in order to use the capabilities of cluster and parallel machines. However, retargeting also involves changes in other important elements of a VR application, such as devices, content, and interaction techniques. IBM presented similar ideas in its interpretation of Scalable graphics [15], but few many details are available in the paper. In summary, our proposal describes how to retarget devices and interaction techniques in VR applications, as oppose to changes in graphic content only.

2.6 Users of VR Toolkits

The interface of a development environment defines a level of knowledge that users should fulfill. With the exception of Alice [97] and VRML [21], all VR toolkits in Table 2.3 require programming skills from VR developers.

We consider that the design of VR applications should not require strong programming skills. The overall architecture of a VR program and decisions about interaction techniques, devices, and content can be taken without writing code in C++ or Java. For this reason, we differentiate between a VR designer, whose task is to design an application at the architectural level, and a VR developer, whose task is to design the details inside components in the application. InTml allows us to make this distinction. Designers create types of devices, content, or behavior in InTml and plug instances of such types together in applications. Developers fill the details of such types, so each component does what it is designed for. There is a separation of concerns between designers and developers, and a collaboration in the development process.

[9]There is also the Scalable Vector Graphics language, but this refers to a different field.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

Concepts

In this chapter, we describe from different viewpoints our model for Virtual Reality applications. We give first an informal description of the basic concepts, the execution model, and some examples of modeling for devices, behaviors, and media content elements. Second, we give a more in depth description of each concept, the way they are related to each other, and their XML syntax. Finally, we present a more formal description in the Z language [87] that clarifies the semantics of the model and generalizes some of the concepts presented in the two previous presentations.

3.1 An Informal Introduction to InTml

We consider a VR application a data flow of interconnected filters, described in a language called InTml, the Interaction Techniques Markup Language. Filters are the building blocks that describe the standard connections for any of the following entities: input or output devices, interaction techniques, object behavior, animations, geometric objects, and other media objects. Details about gathering information from devices or about object behavior code are described in a lower level of abstraction through the use of programming languages. Also, geometry or other media types related to VR objects are produced in any of the available tools for that purpose, such as Maya [3], 3D Max [2], or Blender [14]. InTml is then an integration language for all elements involved in VR applications. It enables the designer to concentrate on the architecture of the application, without dealing with too many details. As an example, while dataflow–based languages such as VRML focuses on description of geometry and animation, InTml focuses on the integration of application–specific behavior, object behavior, and events from input devices. Geometry is something that is described at a lower level, in a loadable format, and InTml refers to it as a reference to an object. The same can be applied to sound or haptic content.

A *filter* represents any device, interaction technique, behavior, or content in a VR application. Its interface is defined in terms of input and output ports, which are the type of events it can receive or produce, respectively. Some input ports can be considered parameters, or ports that will receive information only once at application startup. A filter can have an internal state, which is important in order to model complex filters. However, we do not include this description at the architectural level due to its low–level nature. Figure 3.1 shows a way to represent a filter, `SelectByTouching`, with input ports on the left of a box and output ports on the right. In this particular example, its output port is a selected object from the scene, and its input ports are the VR object used as hand representation, the current position and orientation of such an object, the scene of objects to pick from, and the events that inform about added or deleted objects from the scene. Note that the input ports for the hand representation and the scene can be considered parameters of such a filter, i.e. they will not change once they are assigned.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

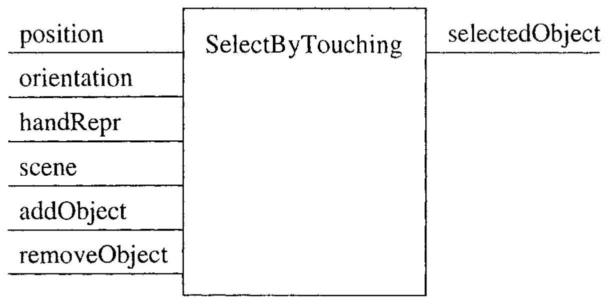


Figure 3.1: Select by Touching. An Example of a Filter

The computation of a filter is divided in three main stages:

- Data collection. All information generated in a certain time interval is collected. This stage is considered a preprocessing stage, in which filters select and manipulate the information they have received, in order to prepare for the next stage.
- Processing. In this stage a filter executes, given the collected input information and its internal state. Output information is generated, but not propagated
- Output propagation. Output information is propagated to all interested filters.

VR objects represent identifiable pieces of content in the virtual environment: elements that can be seen, heard, or touched by the user. An *object holder* is a filter that associates one object to a set of desired changes. It is drawn with an additional decoration for a special input port, object, that receives objects to be hold (a small rectangle between the port and the object holder) [1]. Once an object **O** is associated to an object holder, by sending such an object through the port object, all information coming to the object holder is redirected to **O**. In the same way, outputs from **O** are sent to the filters connected to the object holder. An *application* is a set of interconnected filters, that meet certain user requirements. Figure 3.2 shows a simple application, which allows a user to move a virtual hand with a tracker and touch virtual objects. In this example, a device (`handTracker`) gives position and orientation information to a selection technique (`SelectByTouching`) and an object holder (`handRepresentation`). The actual object representing the user's hand (`handRepr`) is given to `SelectByTouching` for collision detection, and to `handRepresentation` for changing the object. Once a collision is detected, the collided object is passed to `Feedback`, which changes the color of the object. Filters and applications are independent of any particular software framework and hardware, so the designer does not have to be limited by platform specific elements, and the developer is free to reorganize the implementation in order to improve the performance of the application in a particular platform.

[1]For more details about object holders, see Section 3.4.11

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

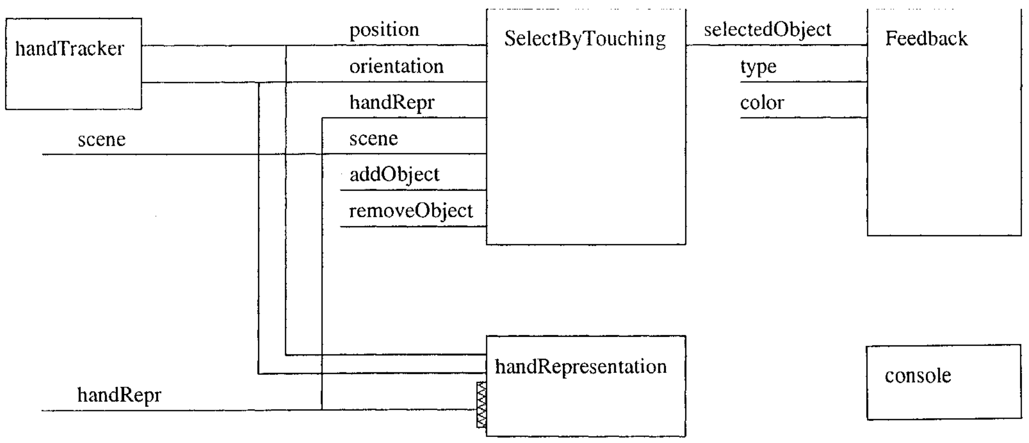


Figure 3.2: Simple Application. Touching Objects With a Virtual Hand.

An *input device* is a filter with just output ports that sends events of a certain type to the dataflow. An *output device* is a placeholder that describes where the output of the application will be displayed – it is internally related to the VR objects, but the details are hidden to the VR designer.

In order to reduce the complexity of an application, subsets of interconnected filters can be encapsulated in a *composed filter*. A composed filter represents a complex behavior in an application, that might be treated as a unit and reused in new applications. Composed filters can be used to encapsulate all necessary details of an interaction technique. As an example, Figure 3.3 shows two views of the Go-Go interaction technique [71] – an interaction technique to lengthen the user's virtual arm for reaching distant objects. The left image shows enough detail to allow VR designers to use such an interaction technique, while the image at the right shows all the filters and objects involved.

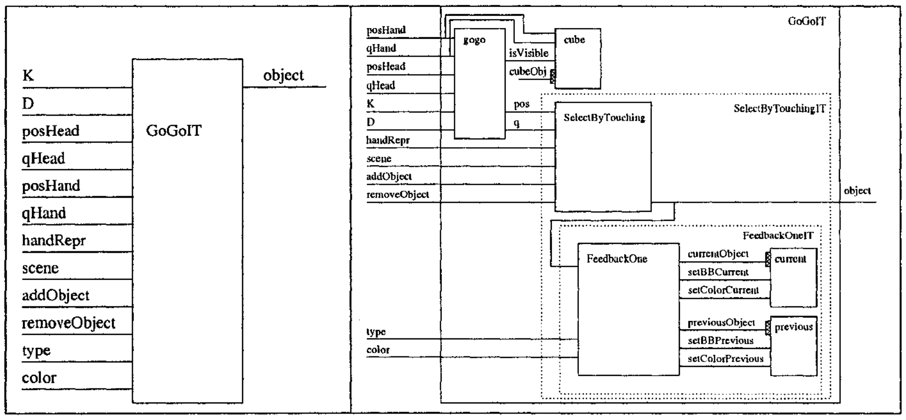


Figure 3.3: The Go-Go interaction technique. General and detailed views.

3.2 Execution Model of an InTml Application

An InTml application is executed as a sequence of identical steps, each one composed of four stages. The actual execution time of each step and the resulting application frame rate are considered implementation dependant, since they vary according to the computational

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

power of the particular hardware platform and the particular method for translating InTml to such a target environment. We assume that a minimum frame rate per device is known, and that the translation process from InTml to code takes into account such rates in order to provide a usable virtual experience. The semantic model of InTml defines which filters are executed in each step, no matter their order, and which results are expected. It is up to the InTml implementation to execute such filters fast enough to reach the target speed of the VR application.

Figure 3.4 shows the stages in an InTml execution step. The activities performed during each stage are the following:

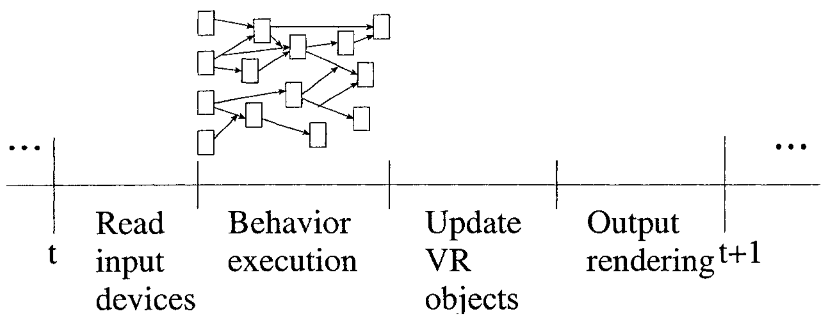


Figure 3.4: Execution Model of InTml Applications between two rendering steps.

- Device reading: Events from active devices are read during this period of time. All events during such a period are considered simultaneous.
- Data flow execution: Events received in the previous stage are fed to the dataflow, which propagates the events throughout reachable filters. All changes to objects requested by filters are queued, and they will be executed in the following stage.
- Object updating: All changes requested in the previous stage are considered, and the selected ones are executed. Each object type should implement its own conflict resolution policy, when several conflicting changes are requested. For example, if an object receives several position changes, it might decide to either execute any of the changes at random, or execute the average of the requested changes. This objects' feature is important due to the fact that the order of execution of filters behind any object is unknown, since InTml implementations with more computational power might execute filters in parallel, and each object might receive changes from more than one filter. However, a selection policy defines if the dataflow is deterministic or not [2].
- Object rendering: Changes to objects are shown to users, in each one of the output devices available. This stage is usually transparent to users of modern graphic APIs, such as Performer and Java3D, and usually supported by dedicated hardware.

The non-deterministic feature of some object updating policies deserves one more comment. While this is in general an undesirable characteristic, created by the availability of more than one input event of a certain type, in practice it is not very noticeable. If we assume that all input events have been generated from a certain device, or by several filters that indirectly received such events, and that filters compute "smooth" functions, the spatial and temporal coherence of the input events will guarantee certain proximity of the values computed from each input event. In practice, when input values are close enough, users will not notice differences related to the chosen value.

[2]The average policy results in a deterministic execution, whereas a random selection policy will produce a non-deterministic result.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Stages can be parallelized or pipelined, so it is possible to get the best performance from each platform, with only one application description. An example is shown in Figure 3.5. Additional threads are dedicated to device event gathering, filter execution, or object updating. There are some points where threads should be synchronized, but in general this approach permits a better application throughput and more complex computations.

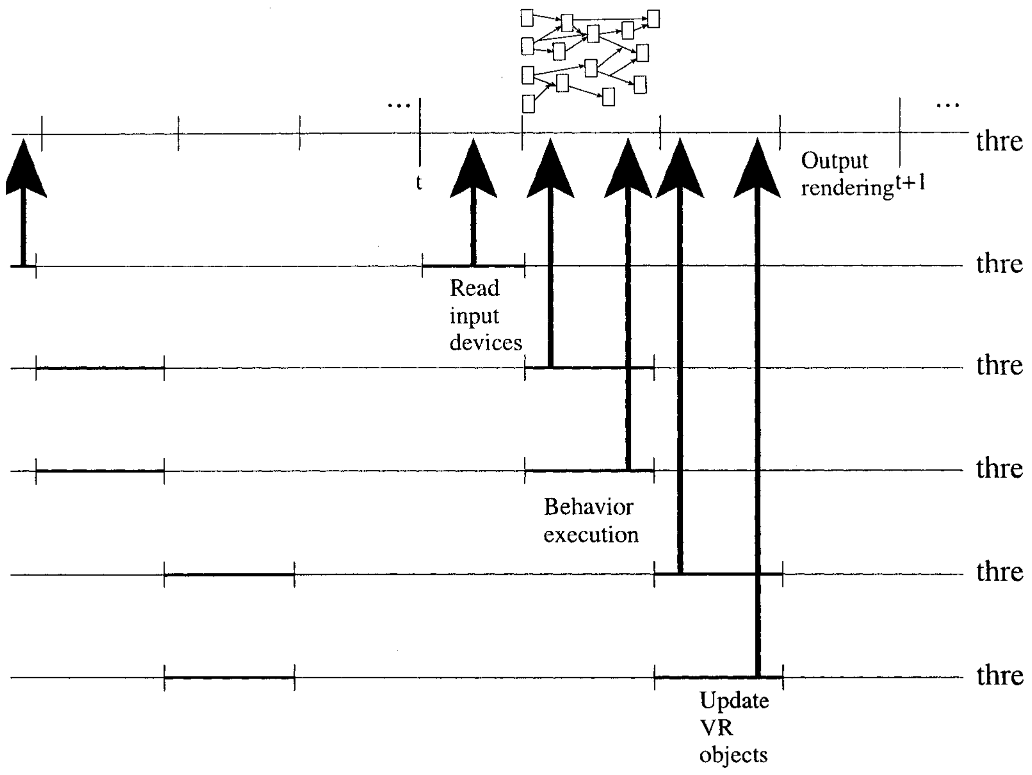


Figure 3.5: Extended InTml Execution Model.

Information in the dataflow and the state of VR objects are considered immutable in a particular time frame. For this reason all filters will see the same state of an object inside a computation frame. An InTml application describes the first two stages — devices to read and data flow —, and the other two are hidden at this level from the designer. We assume an implicit connection between media objects and output devices, in order to allow devices to render the entire state of the world.

From the point of view of designers of VR applications, an InTml application is a set of modules that has to be implemented on top of a foundation framework, and certain rules of execution have to be taken into account. The designer's work is divided between the definition of new filters, reuse of previously defined ones, and definition of applications. Designers collaborate with developers of VR applications, whose main job in an InTml-based environment is to develop the inner code in the filters [3] .

3.2.1 Implementation issues of the Execution Model

As we have mentioned, the actual execution time of a particular InTml application depends on the implementation in which the code is running on, since all concepts related to time are abstracted from the general description we have presented. In particular, we assume that all information generated at a particular time simultaneously arrives to interested filters.

[3]This separation of roles will be described in more detail in Section 4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The purpose of this abstraction is to hide complexity to designers, whose work is to describe a solution no matter the hardware characteristics of the platform they have available. An implementation of InTml has to address the following issues related with time:

- Network delay. This delay is due to the physical characteristics of the network technology used between the computers of a specific platform. Once it is measured, it can be considered constant, no matter the network load.
- Processing delay. This delay is due to the inherent time required to compute results inside a filter. It depends on the amount of input at a particular time, and the algorithms used in the implementation of a filter. It is possible to create models for particular filters in order to predict this delay and take decisions regarding frame execution.
- Latency. This time is caused by the load in the network. It is difficult to predict, but it is possible to define a maximum waiting time, used to discard messages that arrive late.

A simple implementation of InTml, in which there is only one computer receiving data and processing it, does not require to handle the previous concepts. In a more complex setup, it is possible to create a model based on the previous time definitions in order to handle such delays. A simple way to deal with such issues, based on the framework for multi-modal VR applications presented in [95], requires that every filter waits for a period of time before collecting information from input ports. Such a period of time assures that all simultaneous events are received before its processing, so a filter can execute in the normal way without extra considerations.

3.3 InTml Examples

InTml is useful to represent devices, interaction techniques, content behavior, and applications that combine all these elements. Let us illustrate these concepts with some simple examples. A real design of such elements might be different in order to support a more modular and reusable structure, but the presentation here aims at illustrating instead of optimizing for reuse.

One can start with a common input device, a three–button mouse. A graphical representation of a mouse with three buttons in InTml is shown in Figure 3.6. We show the information one can get as separate output ports: x position, y position, both coordinates together as a mouse move, and events for the actions of pressing and releasing each button[4]. Other representations and events are possible. For example, buttons can be represented as separate devices inside a mouse, and events for clicking or double clicking a button can be generated, separate from the press and release events. Any element can have redundant output ports. In this way, other elements can choose what is the right type of information they are interested in.

[4]Types for each event are not shown in this diagram, but each port only allows events of a certain type.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

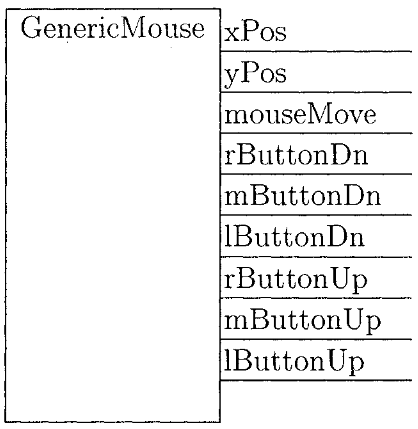


Figure 3.6: A Simple Representation of a Mouse in InTml.

A more advanced device is the 6DOF wand tracker from InterSense. It not only provides position and orientation in 3D space, but it also has a small joystick and four buttons. Such a wand can be represented in InTml as it is shown in Figure 3.7. In this case, buttons only generate one type of event, when they are clicked. It is also possible to have events when a button is pressed or released, as in the mouse example.

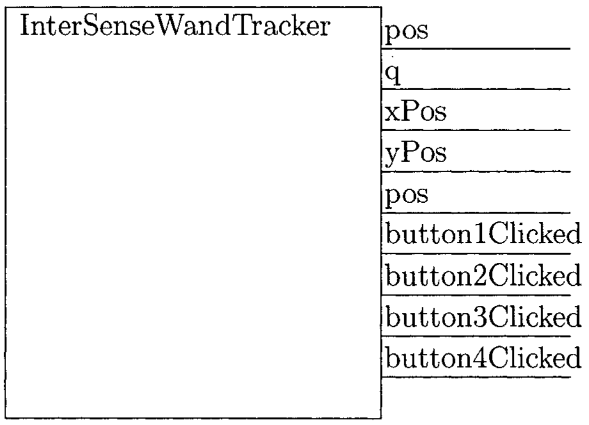


Figure 3.7: An InTml Representation of the InterSense Wand.

An example of a 3D selection widget is the ring menu [58]. A ring menu shows a set of objects in a ring that can be rotated along its 3D axis. Extra geometry in the middle of the ring makes the selected object more visible, which is between the user's viewpoint and the ring's axis. An InTml representation of a ring menu is shown in Figure 3.8.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

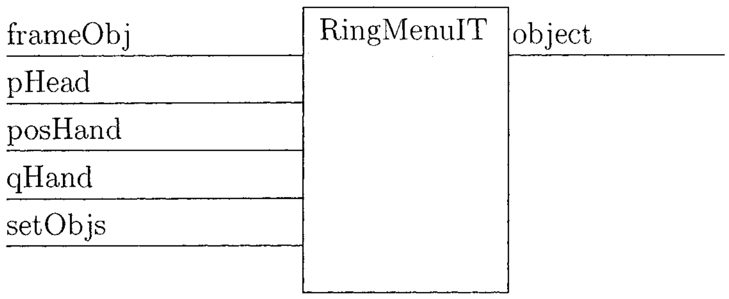


Figure 3.8: An InTml Representation of the Ring Menu.

The geometry for the frame can be given through the `frameObj` port. The position of the user's head is also required, with the position and orientation of a hand tracker. Its output is the selected object, if it has changed since the last selection.

Barrilleaux [8] describes several interaction techniques for 3D manipulation in standard desktops. For example, object movements can be performed in relation to the displacement of the mouse in the display plane, or the projetion of such a movement over the virtual "floor". The latest option is called world–related–movement, and it can be represented in InTml as it is shown in Figure 3.9. The manipulation technique receives a position `p` in display coordinates, the plane that represents the floor in the current scene, the object to be moved and the current user's viewpoint. The result is a new 3D position for the object.

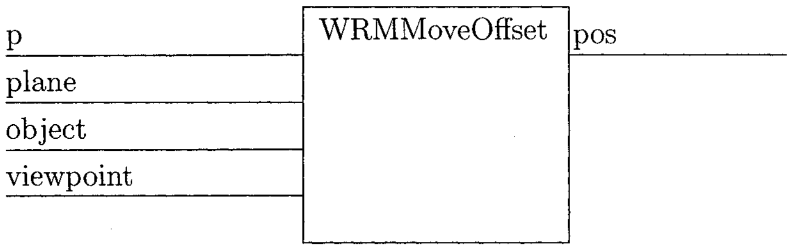


Figure 3.9: An InTml Representation of World–Related Movement.

The basic behavior for objects in the world can also be represented in InTml. Figure 3.10 shows `VRObject`, which encapsulates basic behavior of an interactive object. Position, orientation, and scale can be changed, in a separate way or at once by a transformation matrix. Parts can be added and removed, a bounding box can be drawn around the object, and its main color can be changed. An object can also inform any interested filter of any of the previous modifications.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

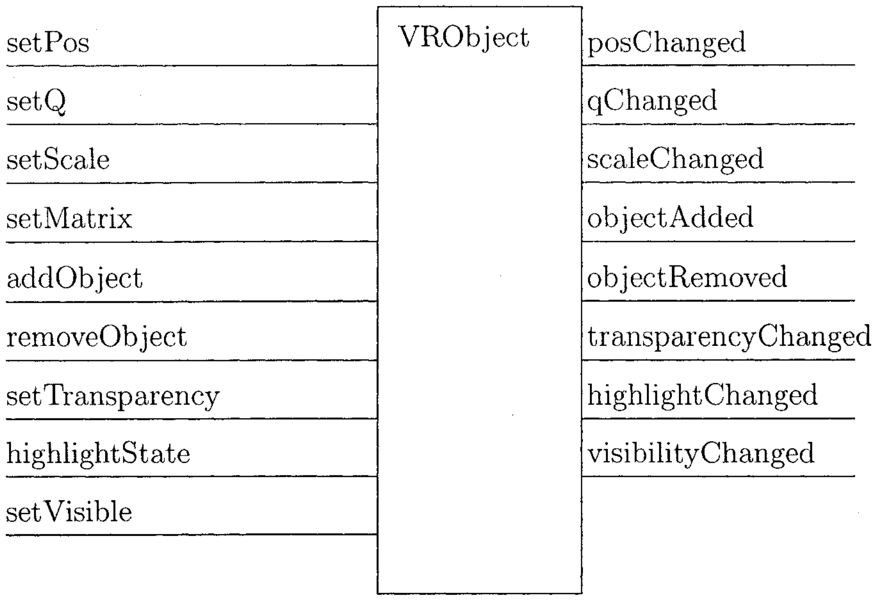


Figure 3.10: An InTml Representation of an Interactive Object.

Examples of applications will be given in Section 5.

3.4 InTml Ontology and XML representation

We describe in this section the concepts in InTml and their relationships, together to the XML syntax we have created for them. Figure 3.11 shows two views of the concepts (in rectangles) and relationships (as arrows) in InTml. A dashed box represents an abstract concept with no instances, but useful to describe relationships of several other concepts related to it by an `isA` relationship. The following paragraphs describe these concepts and relationships in more detail [5]. This description is related to the current XML implementation, according to the DTD definition presented in Appendix C. The description in Section 3.5 separates from current implementations and describes a more general approach, with more solid semantics. XML fragments in this section use the following conventions: String values are written as `value`, optional attributes or entities are enclosed in "[ ]", optional values are written as *opt1/opt2/opt3*, where *opt1* is the value by default.

[5]Section 6 describes the coverage of this concepts in the platforms where InTml has been implemented.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

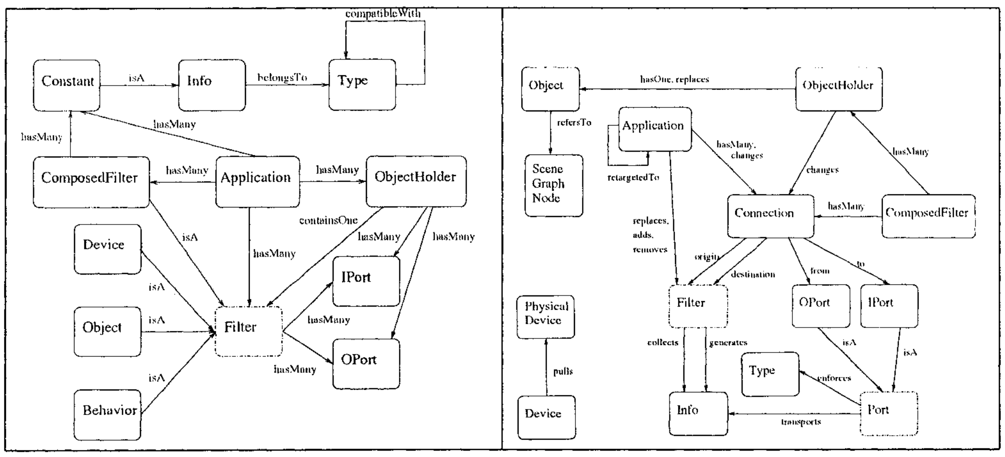


Figure 3.11: Entities and relationships in InTml.

3.4.1 Port

A port is a filter's connection point. It is called an input port (iport) if receives information, and an output port (oport) if propagates information generated from the filter.

The main purpose of a port is to transport information from one filter (origin) to another (destination), which are connected by an oport and an iport, respectively. The type of information flowing through a port should be compatible with the port type.

We have not designed any XML representation for this abstract entity.

3.4.2 IPort

An input port is an entry point for information of a certain type. Its declaration inside a filter class has the following syntax:

```
<IPort id="aName" type="aType" [defValue="stringRep"] [policy="aPolicyName"]
      [isArray="false/true"] [typeArray="static/dynamic"]
      [maxArray="aNumber"] >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
</IPort>
```

An input port is identified by a name. The information received by the port should have a compatible type with the input port's type. An input port might have a default value, which corresponds to the first value received. The attribute `policy` refers to the name of the policy management for multiple events of the same type at the same time. It is optional, with implementation–dependant values. It is also assumed that any implementation has one policy by default [6]. As a way to reduce the syntax of some filter classes, and also as a way to create multiplexors and mergers, we use the following attributes to describe an array of ports with the same basic syntax. The attribute `isArray` is true when the node is an array of input ports, in which case the next two attributes can be defined: `typeArray` that says if there is a fixed (static) or variable number of ports in the array, and `maxArray` that defines the maximum number of ports if the array is static. For example, an IPort declaration such as

[6]For example, in the Java3D implementation described in Section 6 we use two names: ALL and ANY to describe a policy that takes into account all events, or one at random, respectively.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<IPort id="ip1" type="int" isArray="true" typeArray="dynamic"/>
```

will allow references to ports `ip1[2]` or `ip1[15]`, whereas

```
<IPort id="ip1" type="int" isArray="true" typeArray="static" maxArray="10"/>
```

will allow just the first one.

Any input port might have a short and a long description. It can also be associated to an object holder, in which case it is not explicitly declared (i.e. there is no declaration of a type for an object holder), but it is deduced from the type of the first object connected to the object holder [7].

References to ports are used inside connections. This syntax will be described later in this chapter.

3.4.3 OPort

An output port is a channel of information from a filter. A filter can have several output ports, some of them redundant [8], with an entire set of information it can produce. Its XML structure is similar to the one of an input port:

```
<OPort id="aName" type="aType"
       [isArray="false/true"]  [typeArray="static/dynamic"]
       [maxArray="aNumber"]  >
  <ShortDesc></ShortDesc>      [<Description></Description>]
</OPort>
```

The attributes `id`, `type`, `isArray`, `typeArray`, and `maxArray` have the same meaning as the ones in an IPort. Output ports do not have a `defValue`, since the only information that goes out of a filter is the one computed from the information in its input ports, at any period of time. Neither does it have a `policy`, since all information generated is sent to interested filters, and it is up to their input ports to decide a policy for several simultaneous events. In the same way as an IPort, it may have some textual description associated and it may be associated to an ObjectHolder.

3.4.4 Filter

A Filter is the minimum unit of computation in InTml. It defines a set of input ports that receive events from other filters, and a set of output ports that sends computed values from the received events at any time. It also can have an internal state, hidden from the external definition in terms of ports [9]. A filter can be part of composed filters, applications, or object holders. It is involved in connections by binding an output port of a filter with an input port of another. There are three concrete flavors of filters: devices, objects (or media), and behavior (or interaction techniques). It performs the following three tasks at every execution step: collection of events from its input ports, processing of this information, and generation of events to be send through output ports. It does not directly have instances, but the ones from its subclasses (Device, Object, Behavior).

3.4.5 Device

A device [10] is the simplest filter class. It represents a physical unit that produces certain type of information. It usually gets its information by pulling it from a physical device,

[7]More details in Section 3.4.11.

[8]In some cases it is useful to count with several ports with several representations of the same information, i.e. a matrix and a quaternion representation of a rotation.

[9]Such an internal state is usually documented for completeness reasons

[10]Sometimes is called logical device, in contrast to a physical device, defined below.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

but the relationship is not necessarily one to one, i.e. a logical device can pull information from several physical devices, or a physical device can be associated to several logical ones. Devices are instances of device classes, declared as follows:

```
<DeviceClass id="aName" >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
  [<Implements classId="aClassName"/>]
  [<IPort>...</IPort>]
  [<OPort>...</OPort>]
  ...
</DeviceClass>
```

A device class has a name as identifier, and it contains the input and output ports of every instance of such a class. The special tag `Implements` allows an inheritance–like relationship between device classes: The class contains all input and output ports of the implemented one. An instance of a device class can be created inside an application as follows

```
<IDevice id="aName" type="aType"/>
```

or

```
<ODevice id="aName" type="aType"/>
```

The IDevice emphasizes the fact that the device will produce input information for other filters in the application. The purpose of ODevice is to mention output devices in the environment. If a device both produces and consumes information at the same time, it should be considered as an IDevice.

3.4.6 Physical Device

A physical device is a physical entity that produces information to be read by the computer. It is represented in an InTml application as one or many instances of class `Device`. Special tuning of physical devices (callibration procedures, startup, alignment, particular method for accessing information from it) are considered out of the scope of InTml. In this way, InTml users concentrate on what type of information they can use from devices, instead of both information and setup. There is no visibility of physical devices in InTml, appart from a tacit correspondence with logical devices.

3.4.7 Object

An object is a filter that affects the media involved in the VR application, i.e. a piece of geometry, a sound effect, a haptic effect, etc. In the case of geometry, media is usually represented with a scene graph. InTml objects are built on top of scene graph nodes, and are used inside an InTml application to make changes in particular nodes in the scene. During an execution step, an object queues all changes that filters request. At the end of the execution step, once all filters involved in its input have been executed, an object executes the requested changes according to the policies of its input ports. If the object report changes through its output ports, they will be noticed in the next execution step. In this way it is assured that all filters can read the same state of an object during one execution step, no matter the order of execution. An object type is defined as follows:

```
<ObjectClass id="aName" >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  [<Implements classId="aClassName"/>]
  [<IPort>...</IPort>]
  [<OPort>...</OPort>]
  ...
</ObjectClass>
```

This structure is identical to the one for device classes. An instance of an object can be created inside an application or a composed filter as follows:

```
<Object id="aName" type="aType" [fileName="aFile"]
        [primitive="Box/Cone/Cylinder/Ellipse"] />
```

Every object has an unique identifier (aName), declares the name of its object class (aType), can load a piece of geometry or other loadable type of content from a file (aFile), or it can correspond to one of 4 simple shapes (*Box/Cone/Cylinder/Ellipse*).

Objects can also be sent to other filters as information, and in particular to object holders. The following syntax is used for this task:

```
<Binding iE="_self" iP="objectName"
        oE="aFilter" oP="anIPort" />
```

The special identifier _self refers to the current composed filter or application that contains the object called objectName. The filter aFilter will receive objectName through the port anIPort. Objects can be plugged to object holders through a special port called object.

3.4.8 Scene Graph Node

A Scene graph is the main data structure for geometry processing in modern graphic APIs, such as Performer [79] or Java3D [63]. An object in InTml can refer to an element in such a hierarchy, in such a way that it hides the complexity of geometry rendering inherent to the scene graph structure but at the same time allows interaction with such elements. Figure 3.12 shows the relationship between nodes in the scene graph and InTml objects. There could be an implicit dependance between InTml objects, since they can refer to scene graphs nodes that depend on each other, i.e. objects A and B in the figure. There are ways to make this relationship explicit, for example, by allowing InTml objects to replicate such a hierarchy. However, in the general case, we consider this relationship out of the scope of the InTml description.

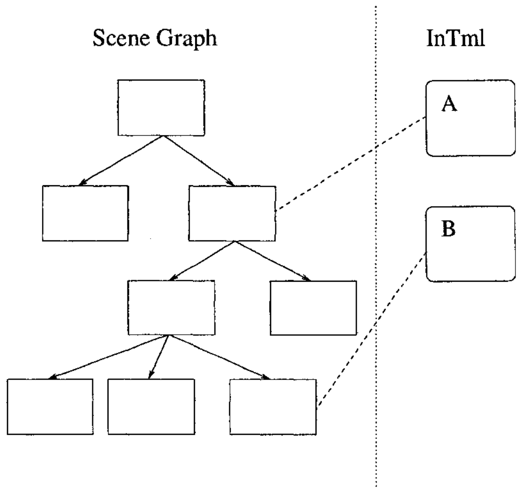


Figure 3.12: Relationship between scene graph nodes and InTml objects.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4.9 Behavior

Behavior is represented by filters in InTml. A filter can represent a piece of computation in an InTml application. It represents interaction techniques, special object behavior, animations, or application specific behavior as filters in the dataflow. In its simplest form, the type of a filter in InTml is declared as follows:

```
<FilterClass id="aName" >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
  [<Implements classId="aClassName"/>]
  [<IPort>...</IPort>]
  [<OPort>...</OPort>]
  ...
</FilterClass>
```

This has the same structure of a DeviceClass or an ObjectClass. Filter instances can be created in applications or composed filters with the following statement:

```
<Filter id="aName" type="aType"/>
```

As in devices and objects, a filter is identified by a name and a type.

3.4.10 Connection

A connection defines a relationship between devices, behavior, and media content. It is the only way to pass information from one software component to another, and the only visible relationship possible at runtime [11]. The XML syntax of a connection is as follows:

```
<Binding iE="aFilter" iP="anOPort" [iI="anIndex"]
         oE="aFilter" oP="anIPort" [oI="anIndex"] />
```

A connection is uniquely identified by an origin element (`iE`), an output port in such an element (`iP`), an optional index in the output port declaration (`iI`), a destination element (`oE`), an input port in such an element (`oP`), and an optional index for the input port (`oI`). Connections appear inside applications and filter classes in order to describe the dataflow between elements.

3.4.11 ObjectHolder

An object holder is a way to define placeholders with a certain structure of connections in a dataflow, places where media content elements can be plugged, executed, and changed. Section 3.5 will describe a more general concept, a filter holder, allowed to hold any type of element. Current implementations of InTml are limited to hold just objects, but this constraint can be avoided in the future, by following the semantics of filter holders [12].

The XML syntax for an object holder is as follows:

```
<ObjectHolder id="aName" />
```

We can notice that an object holder does not have a type in its declaration. Our current implementation of object holders finds out its ports by "copying" the ports of the first object it gets to hold. Additionally, an object holder will always have two additional ports: an iport called object that allows receiving new objects, and an oport called objectChanged that will inform interested filters about changes in the contained object. If such an object is replaced, subsequent objects will be connected to the ports already defined at the object holder, provided that ports are compatible [13]. An example of this mechanism is shown in

[11]Devices are related to physical devices and objects to scene graph nodes, but such relationships are considered out of the scope of InTml.

[12]Filter holders are defined later in this chapter.

[13]Currently, compatibility between ports is defined by their names.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 3.13. When the application starts and an object holder has not received any events, there are only two ports declared: one for receiving object events, one to inform changes in the contained object. Once an object `A` is connected, the object holder copies its definition and adds to itself the corresponding ports. Later on, when `A` is replaced by an object `B` of a different class, just the compatible ports are connected, i.e. ports `r` and `s` will not be connected. If `A` is embedded again in the object holder, it will be conneted as it is was before.

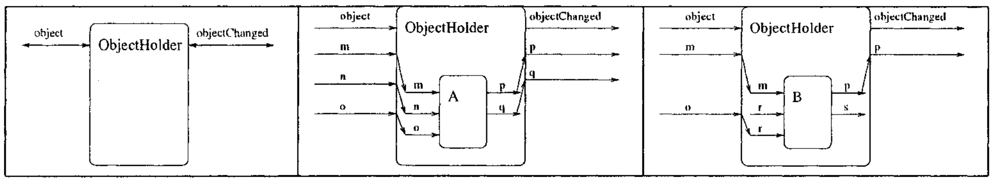


Figure 3.13: Different states of an Object Holder during execution.

3.4.12 Events in the Dataflow and Types

Every piece of information flowing through the dataflow is an event, or *Info* in Figure 3.11. Pieces of information have a type associated to it and its value could be atomic or composed. A value is considered atomic if it is implementation–dependant and do not have a complex representation in InTml. Note that a complex structure such a quaternion [73] can be treated as atomic if there is no definition of its structure in InTml. A value is composed if they correspond to objects defined in InTml. Types can have an implicit compatibility relationship, i.e. inheritance in object–oriented languages. However, this relationship is not declared in the current implementation at the level of InTml, but at the implementation level. Type declarations are either implicit, when they appear as types of ports or constants, or explicit, when they are described as an `ObjectClass`.

All events are time–stamped, and all events received at the same frame from devices and propagated through the dataflow are considered simultaneous. Events are immutable, so their values can not be changed by filters. An event can be propagated to several filters (fan-out), so several filters can refer to the same Info.

3.4.13 Constant

A constant is a value of a certain type. Constants are used to give a default initial value to a filter, through a particular input port. A constant is declared as follows

```
<Constant id="aName" type="aType" value="aValue"/>
```

A constant has an identifier and a type. Its value is a string–based representation of its value inside the InTml execution. Constants can be pushed through input ports, so its value will be the first value received by a port. Constants can be declared inside applications and composed filters, and sent to filters through a particular input port. The following is an example, in which a filter `f` receives a constant `c1` through its input port `ip`. We use the special identifier `_self` in order to refer the constant `c1` defined in the current context.

```
<Binding iE="_self" iP="c1" oE="f" oP="ip" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4.14 ComposedFilter

A composed filter is an InTml construct that allows designers to hide complexity by encapsulating a piece of an InTml program as a simple filter in the environment. A composed filter describes a subset of filters, objects, object holders, constants, and connections that execute certain tasks. We use the same XML element to declare simple and composed filters, but a composed filter can include other elements, such as:

```
<FilterClass id="aName" >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
  [<Implements classId="aClassName"/>]
  [<IPort>...</IPort>]
  [<OPort>...</OPort>]
  ...
  [<Filter>...</>]
  [<Object>...</>]
  [<ObjectHolder>...</>]
  [<Constant>...</>]
  [<Binding>...</>]
  ...
</FilterClass>
```

A composed fiter describes a dataflow with a certain interface given by its own input and output ports. IPort and OPort elements should be connected to ports of internal entities through Binding statements. For example:

```
<FilterClass id="ComposedFilter1" >
  <ShortDesc>An example of composed filter</ShortDesc>
  <IPort id="ip1" type="Type1"/>
  <OPort id="op1" type="Type1"/>
  ...
  <Filter id="f1" type="Filter1"/>
  <Object id="obj1" type="Object1" fileName="f1"/>
  <Binding iE="_self" iP="ip1" oE="f1" oP="ip"/>
  <Binding iE="obj1" iP="op" oE="_self" oP="op1"/>
  ...
</FilterClass>
```

Assuming that filter f1 has an input port ip, and that object obj1 has an output port op, the previous example connects the input and output of the composed filter to its internal structure. Note the use of _self to refer to the composed filter, when necessary.

An instance of a composed filter can be created inside an application or inside other composed filters with the same syntax used for normal behavior.

3.4.15 Application

An application element in InTml describes a dataflow of filters that accomplish certain tasks. The XML syntax for an application is the following:

```
<App id="aName" >
  <ShortDesc></ShortDesc>
  [<Description></Description>]
  ...
  [<IDevice>...</>]
  [<ODevice>...</>]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  [<Filter>...</>]
  [<Object>...</>]
  [<ObjectHolder>...</>]
  [<Constant>...</>]
  [<Binding>...</>]
  ...
</App>
```

An application is identified by a name. It contains all the required elements for a dataflow in InTml: input devices, output devices, filters, objects, object holders, constants, and connections.

Two important development tasks are involved with applications: initial definition, and retargeting to other hardware platforms. The initial definition creates the required filter classes and connections to satisfy certain set of user requirements in a particular platform. A retargeting process consists of deriving new applications from a previously defined one, by replacing elements for the more suitable ones in a new hardware platform. Such a change starts in devices, and changes can be propagated to filters, constants, objects, object holders, and connections.

3.4.16 Other Language Features

The XML representation of InTml has some extra constructs, which are useful at design time or for documentation purposes. These concepts are the following: `Package`, `Import`, `Overrides`, `Platform`, `Index`, and `PaperRef`.

`Package` allows the creation of name spaces in InTml. The name of a filter class can be used without qualifiers by classes or applications of the same package. Otherwise it should be fully qualified, or its package should be imported. The sentence `Import` allows any package to refer to filter classes in a non fully–qualified form. The syntax for these two elements is as follows:

```
<Package id="aName">
  [<Import id="aPackageName">]
  ...
  [<DeviceClass>...</DeviceClass>]
  [<FilterClass>...</FilterClass>]
  [<ObjectClass>...</ObjectClass>]
</Package>
```

A package gives a prefix to all classes declared inside of it. Classes can be used without a qualifier inside classes of the same package. Outside the package, classes can be named either by its fully qualified name, or by importing its package. When an package is imported, all its classes can be used with their simple names. Conflict names in the current implementation are resolved by taking the first match to a filter class in the list of imported packages.

An application is not defined inside a package declaration, but instead it can be created with its fully qualified name. For example:

```
<App id="a.b.c" >
  ...
</App>
```

defines application `c`, in the package `a.b`.

`Overrides` is a special relationship between applications. If application `B` overrides `A`, the resulting application is the set operation $(A \setminus B) \cup B$. In other words, `B` has all objects, behaviors, devices, and connections in `A` that are not defined in `B`, plus all elements of `B`. In this way, `B` replaces elements in `A` with new definitions. This mechanism has been used in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

order to create default values for an application: A corresponds to the basic implementation of an application, common to other ones, while B redefines only some elements and adds many more.

Platform is a special construct that groups together sets of devices. It is planned to be used for automatic retargeting of applications, but their semantics is not fully defined.

Index in an extra tag that applications and class declarations have for documentation purposes. Index classifies an application or a class under an index name. Several indexes are allowed, so an element can be classified under several criteria. For example, the following declaration:

```
<FilterClass id="SelectByTouching">
  <Indexes>
    <Index id="first" value="intml.selection.details"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  ...
</FilterClass>
```

classifies the class SelectByTouching under two criteria: first, with value intml. selection.details, and papers with the special value _hidden, which is used to officially hide an element from a particular classification. Documentation tools described in Section 6 will use this information to create browsers of elements.

3.5 A Formal Description of InTml in the Z Language

A Virtual Reality application is described here as a flow of messages in which input messages or events are read from devices and propagated to all the functionality of an application, represented in terms of filters. This dataflow might be executed in parallel or pipelined, complex functions can be encapsulated in order to reduce complexity, information from several simultaneous devices can be received, and all simultaneous filters are guaranteed to see the same world state. A formal description of this architecture based on the Z language [87] is given here, and it is shown how these special characteristics are accomplished, independently from a particular implementation [14]. Such a formal description serves as a blueprint for new implementations of this architecture, a reference that explains the semantics of InTml, independently from any particular implementation. Finally, we analyze the differences between our presentation and theirs, and some of the lessons learned from this specification exercise.

3.5.1 Basic Concepts

A *filter* is a processing element in the dataflow. Names for filters are unique in the system, and belong to the set L. Filters are connected by channels, whose names are also unique and belong to the set C. A *channel* uniquely describes a particular output of a filter [15]. *Messages* are indivisible chunks of information that go through channels and belong to the set M.

A *stream* is a function that relates a channel name with a particular sequence of messages. Messages in a stream are ordered in bags, and each bag represents a set of messages received in a particular interval. In this way, more than one message can be received in a particular interval, messages of different intervals can be identified, and the order of messages in the same interval is not relevant, a very useful property for non-synchronous, parallel execution

[14]A brief introduction to the Z language and the concepts used in this specification can be found in Appendix F.

[15]Channels are a concept defined by [68] which facilitates connections between filters. It replaces at the specification level the concept of ports, which are the entry and exit points of channels.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of components. We use the property described by Philipps and Rumpe [68] that it is impossible to distinguish between a function that computes its output given the history of its input so far, and one that selects the current output from a sequence of precomputed results. This gives uniformity between our presentation and theirs in order to reuse some of their specification styles.

$$\begin{array}{|l}
\textit{Stream} \\
\hline
c : C \\
m : \mathrm{seq}(\mathrm{bag}\, M) \\
\hline
\end{array}$$

We use the operator $\downarrow$ [16] to refer to the messages in a stream up to a certain moment, and it is defined in several contexts. In its simple form, $st \downarrow i$ is the operation of extracting a sequence from a stream st with only the first i elements.

$$\begin{array}{|l}
[X] \\
\hline
\_ \downarrow \_ : \mathrm{seq}\, X \times \mathbb{N} \longrightarrow \mathrm{seq}\, X \\
\hline
\forall\, st : \mathrm{seq}\, X;\ i : \mathbb{N} \bullet \\
st \downarrow i = (1 \mathinner{\ldotp\ldotp} i) \lhd st \\
\hline
\end{array}$$

We also use this operator at the level of streams

$$\begin{array}{|l}
\hline
\_ \downarrow \_ : \textit{Stream} \times \mathbb{N} \longrightarrow \textit{Stream} \\
\hline
\forall\, st, stO : \textit{Stream};\ i : \mathbb{N} \bullet \\
(st \downarrow i = stO \Leftrightarrow stO.c = st.c \land stO.m = st.m \downarrow i) \\
\hline
\end{array}$$

And at the level of sets of streams

$$\begin{array}{|l}
\hline
\_ \downarrow \_ : \mathbb{P}\, \textit{Stream} \times \mathbb{N} \longrightarrow \mathbb{P}\, \textit{Stream} \\
\hline
\forall\, stSet : \mathbb{P}\, \textit{Stream};\ i : \mathbb{N} \bullet \\
stSet \downarrow i = \{ stI : stSet \bullet stI \downarrow i \} \\
\hline
\end{array}$$

The function *stream* describes the association between a channel name and its corresponding stream. The function *streams* applies to a set of channel names and gives us a set of streams.

$$\begin{array}{|l}
\hline
stream : C \longrightarrow \textit{Stream} \\
streams : \mathbb{P}\, C \longrightarrow \mathbb{P}\, \textit{Stream} \\
\hline
\forall\, c : C \bullet (\exists\, st : \textit{Stream} \bullet stream(c) = st \land st.c = c) \\
\forall\, cSet : \mathbb{P}\, C \bullet streams(cSet) = \{ c : cSet \bullet stream(c) \} \\
\hline
\end{array}$$

We use the function *restr* later, in the definition of a filter function. It is the selection of a subset of streams with specific names.

$$\begin{array}{|l}
\hline
restr : (\mathbb{P}\, \textit{Stream} \times \mathbb{P}\, C) \nrightarrow \mathbb{P}\, \textit{Stream} \\
\hline
\forall\, iSet : \mathbb{P}\, \textit{Stream};\ cSet : \mathbb{P}\, C \bullet \\
restr(iSet, cSet) = \{ s : \textit{Stream} \mid s \in iSet \land s.c \in cSet \} \\
\hline
\end{array}$$

[16]The symbol $\downarrow$ is read "downarrow".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.5.2 Filters and Delays

A *primitive filter* is a computation unit that receives some information in its input streams and computes information each interval in its output streams. By definition, input and output streams do not form cycles. We define the function inside a filter in the schema *Behavior*, with the property that the output of two streams that are equal up to a interval i is the same up to such an interval. In this way, the computation of the behavior up to this interval does not depend on future states, just previous states. Such a function is capable of computing the information in O, the output channels of interest.

Behavior

$$\begin{array}{l} \mathit{fun} : \mathbb{P}\ \mathit{Stream} \leftrightarrow \mathbb{P}\ \mathit{Stream} \\ \hline \forall\, SI, SO : \mathbb{P}\ \mathit{Stream} \bullet \mathit{fun}(SI) = SO \Rightarrow SI \cap SO = \varnothing \\ \forall\, S1, S2 : \mathbb{P}\ \mathit{Stream};\ i : \mathbb{N} \bullet \\ S1 \downarrow i = S2 \downarrow i \Rightarrow \mathit{fun}(S1) \downarrow i = \mathit{fun}(S2) \downarrow i \end{array}$$

PrimitiveFilter

$$\begin{array}{l} \mathit{name} : L \\ I : \mathbb{P}\ C \\ O : \mathbb{P}\ C \\ \mathit{Behavior} \\ \hline \mathit{restr}(\mathit{fun}(\mathit{streams}(I)), O) = \mathit{streams}(O) \\ I \cap O = \varnothing \end{array}$$

A *delay* of one unit is a special type of filter, with one input and one output stream, in which the output at interval i+1 is equal to the input at interval i. We define it as:

Delay

$$\begin{array}{l} \mathit{PrimitiveFilter} \\ ID : C \\ OD : C \\ \hline \{ID\} = I \wedge \{OD\} = O \\ \mathit{fun}(\mathit{streams}\{ID\}) = \mathit{streams}(\{OD\}) \wedge \\ \mathit{tail}\,((\mathit{stream}(OD)).m) = (\mathit{stream}(ID)).m \end{array}$$

We can simulate memory inside a filter by having a delay between a pair of input - output streams, in which the information through the delay can be as complex as necessary:

FilterWithMem

$$\begin{array}{l} \mathit{PrimitiveFilter} \\ \mathit{delay} : \mathit{Delay} \\ \hline \mathit{delay}.ID \in O \wedge \mathit{delay}.OD \in I \end{array}$$

We can also define parameters, as those input channels that just receive information at the beginning of the execution.

FilterWithParams

$$\begin{array}{l} \mathit{PrimitiveFilter} \\ \mathit{params} : \mathbb{P}\ C \\ \hline \mathit{params} \subseteq I \\ \forall\, p : \mathit{params} \bullet (\forall\, i : \mathbb{N} \bullet i > 1 \Rightarrow ((\mathit{stream}(p)).m)(i) = \varnothing) \end{array}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We define a *filter* as a primitive filter that might have memory and parameters,

$$Filter \mathrel{\widehat{=}} PrimitiveFilter \land FilterWithMem \land FilterWithParams$$

3.5.3 An Example: A One-Bit Adder

As an example of how a PrimitiveFilter represents an operation, lets describe an adder of the information from two streams. A one-bit adder has three input channels, two for actual values and one for a carry value, and sends its output through two channels, one for the actual addition and one for the possible carry value. All messages are assumed to be either 1s or 0s. Since an input channel can receive several messages at once, we define the functionality in terms of an appearance of at least a 1 value.

$$\begin{array}{|l}
asBin : M \longrightarrow \{0,1\} \\
asBinBag : \text{bag}\, M \longrightarrow \{0,1\} \\
\hline
\forall\, b : \text{bag}\, M \bullet (asBinBag(b) = 1 \Leftrightarrow (\exists\, m : M \bullet m \sqsubseteq b \land \\
asBin(m) = 1))
\end{array}$$

We use the following definition of a xor function between two bits,

$$\begin{array}{|l}
xor : (\{0,1\} \times \{0,1\}) \longrightarrow \{0,1\} \\
\hline
\forall\, a, b : \{0,1\} \bullet (xor(a,b) = 0 \Leftrightarrow (a = 0 \land b = 0) \lor (a = 1 \land b = 1))
\end{array}$$

A bit adder is then defined as follows:

$$\begin{array}{|l}
\textit{BitAdder} \\
\hline
PrimitiveFilter \\
i1 : C \\
i2 : C \\
cI : C \\
o : C \\
cO : C \\
\hline
I = \{i1, i2, cI\} \land O = \{o, cO\} \\
\forall\, i : \mathbb{N} \bullet \\
\#(((stream(o)).m)(i)) = 1 \land \#(((stream(cO)).m)(i)) = 1 \\
\forall\, i : \mathbb{N} \bullet \\
asBinBag(((stream(o)).m)(i)) = \\
xor(xor(asBinBag(((stream(i1)).m)(i)), asBinBag(((stream(i2)).m)(i))), \\
asBinBag(((stream(cI)).m)(i))) \\
\forall\, i : \mathbb{N} \bullet asBinBag(((stream(cO)).m)(i)) = 1 \Leftrightarrow \\
[\![asBinBag(((stream(i1)).m)(i)), asBinBag(((stream(i2)).m)(i)), \\
asBinBag(((stream(cI)).m)(i))]\!] \sharp\, 1 > 1
\end{array}$$

Such an adder can be used as a parallel or a sequential adder. Several bit adders can be connected in order to add several bits at once, or just one adder can sequentially receive several bits to be added, providing some delayed feedback of the carry information.

3.5.4 Applications

We compose groups of filters to form applications. We define first two auxiliary functions: *successors*, and *paths*. The *successors* of a filter f given a set of filters *iSet* are all those filters in *iSet* that have an input channel connected to an output channel of f,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$succ1 : (\mathbb{P}\,Filter \times Filter \times C) \longrightarrow \mathbb{P}\,Filter$
$successors : (\mathbb{P}\,Filter \times Filter) \longrightarrow \mathbb{P}\,Filter$

$\forall\, iSet : \mathbb{P}\,Filter; f : Filter \bullet (\forall\, c : f.O \bullet$
$succ1(iSet, f, c) = \{f2 : iSet \mid c \in f2.I\})$

$\forall\, iSet : \mathbb{P}\,Filter; f : Filter \bullet$
$successors(iSet, f) = \{f2 : iSet \mid (\exists\, c : C \bullet c \in f.O \land$
$c \in f2.I)\}$

The *paths* from a filter f in a set of filters is the set of all sequences of consecutive successors, starting at f. This set might be infinite, if there are cycles.

$paths : (\mathbb{P}\,Filter \times Filter) \longrightarrow \mathbb{P}(\mathrm{seq}\,Filter)$

$\forall\, iSet : \mathbb{P}\,Filter; f : Filter \bullet$
$paths(iSet, f) = \{s : \mathrm{seq}\,Filter \mid s(0) = f \land (\forall\, i : \mathbb{N} \bullet$
$s(i+1) \in successors(iSet, s(i)))\}$

Given a set of filters F, we say that there is a cycle from one of its filters if it exists a path that name such a filter more than once. We define the function *cyclic* over a set F that gives us a subset of filters with this property.

$cyclic : \mathbb{P}\,Filter \longrightarrow \mathbb{P}\,Filter$

$\forall\, iSet : \mathbb{P}\,Filter \bullet$
$cyclic(iSet) = \{f2 : iSet \mid (\exists\, p : paths(iSet, f2) \bullet$
$(items\, p \sharp f2) > 1)\}$

The concept of cycles is used now for the definition of a composition of filters. A set of interconnected filters create a *composed filter* [17]. A composed filter is uniquely identified by a name, its input and output channels are disjoint, its filters do not share output channels, and filters do not have loops unless they are mediated by delays. Delays can connect two distinct filters or delays, in any combination, so strings of delays are allowed. Objects represent references to content in the application [18]. We avoid two filters from reading different object states in a particular execution frame by forcing objects to be followed by delays. The behavior of a composed filter is defined as the composition of all behaviors in its filters and delays [19].

[17]A ComposedFilter is also a Filter, when all its details are hidden,so several layers of composition are possible.

[18]Such objects might be interrelated, but any conflicts between operations in different objects are treated with application logic, which may be generic.

[19]The last condition in this schema is redundant, but we want to emphasize the relationship between the composed filter's function and the functions of each filter inside.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ComposedFilter

$name : L$
$filters : \mathbb{P}_1\, Filter$
$delays : \mathbb{P}\, Delay$
$objects : \mathbb{P}\, Filter$
$IU : \mathbb{P}\, C$
$OU : \mathbb{P}\, C$
$I : \mathbb{P}\, C$
$O : \mathbb{P}\, C$
$Behavior$

$objects \subseteq filters$

$IU = \bigcup\{f : filters \bullet f.I\} \cup \bigcup\{d : delays \bullet d.I\}$

$OU = \bigcup\{f : filters \bullet f.O\} \cup \bigcup\{d : delays \bullet d.O\}$

$I = IU \setminus OU$

$O = OU$

$\forall f1, f2 : filters \bullet f1 \neq f2 \Rightarrow (f1.name \neq f2.name \wedge$
$f1.name \neq name)$

$I \cap O = \varnothing$

$\forall f1, f2 : filters \bullet f1 \neq f2 \Rightarrow f1.O \cap f2.O = \varnothing$

$cyclic(filters) = \varnothing$

$\forall\, o : objects \bullet (\forall\, c : o.O \bullet c \notin \bigcup\{f : filters \bullet f.I\})$

$restr(fun(streams(I)), O) = streams(O)$

$\forall\, c : O \bullet ((\exists f : filters;\ l : \mathbb{P}\, IU \bullet$
$stream(c) \in f.fun(streams(l))) \vee$
$(\exists\, d : delays;\ l : \mathbb{P}\, IU \bullet stream(c) \in d.fun(streams(l))))$

Our first attempt to define an *application* is as a composed filter with a special subset of filters, called *devices*. Devices are sources or sinks of information. In general, input channels to sources of information are parameters, and the input of a sink of information is the entire set of objects in the application.

Application0

$ComposedFilter$
$devices : \mathbb{P}_1\, Filter$

$devices \subseteq filters \wedge devices \cap objects = \varnothing$

An application can be seen in some cases as a complex filter, by hiding the extra definition. We do that with the function *app2Comp*

$app2Comp : Application0 \longrightarrow ComposedFilter$

$\forall\, a : Application0;\ c : ComposedFilter \bullet app2Comp(a) = c \Leftrightarrow$
$a.name = c.name \wedge a.filters = c.filters \wedge a.delays = c.delays$
$\wedge\ a.objects = c.objects \wedge a.fun = c.fun$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.5.5 Dataflow Execution

An *ExecutionStep* of a filter is the process of obtaining certain information in the output ports, given the information in the input ports at a particular interval. The operation just shows the information in the output channels at a given interval.

ExecutionStep

$\Xi Filter$
$input? : \mathbb{P}(\text{bag } M)$
$output! : \mathbb{P}(\text{bag } M)$
$i? : \mathbb{N}$

$\forall in : input? \bullet (\exists c : I \bullet in = ((stream(c)).m)(i?))$

$\forall out : output! \bullet (\exists c : O' \bullet out = ((stream(c)).m)(i?))$

The initialization state of a filter assigns the input and output channels of such a filter, and executes changes related to its parameters.

InitFilter

$Filter'$
$ic? : \mathbb{P}\, C$
$oc? : \mathbb{P}\, C$
$input? : \mathbb{P}(\text{bag } M)$
$output! : \mathbb{P}(\text{bag } M)$

$I' = ic? \land O' = oc?$

$\forall in : input? \bullet (\exists c : I' \bullet in = ((stream(c)).m)(0))$

$\forall out : output! \bullet (\exists c : O' \bullet out = ((stream(c)).m)(0))$

An execution step for an application is given by the execution step of all filters that have some information in their input ports in a given frame, plus delays that had information in the previous interval. We define the functions *executingFilters* and *executingDelays* to describe which filters and delays are activated at any given interval.

$streamsWithInput : (\mathbb{P}\, Stream \times \mathbb{N}) \longrightarrow \mathbb{P}\, Stream$
$executingFilters : (ComposedFilter \times \mathbb{N}) \longrightarrow \mathbb{P}\, Filter$
$executingDelays : (ComposedFilter \times \mathbb{N}) \longrightarrow \mathbb{P}\, Delay$

$\forall iSet : \mathbb{P}\, Stream; i : \mathbb{N} \bullet streamsWithInput(iSet, i) = \{s : iSet \mid (s.m)(i) \neq \varnothing\}$

$\forall f : ComposedFilter; i : \mathbb{N} \bullet$
$executingFilters(f, i) = \{f2 : f.filters \mid streamsWithInput(streams(f2.I), i) \neq \varnothing\}$

$\forall f : ComposedFilter; i : \mathbb{N} \bullet$
$executingDelays(f, i) = \{d2 : f.delays \mid streamsWithInput(streams(d2.I), i) \neq \varnothing\}$

Is it possible that not all filters and delays are executed in a particular interval, but over time, all filters and delays should be executed. The minimum number of intervals required for the execution of a composed filter is called its minimum execution time.

In order to properly execute an application we define extra requirements. No filter in an application generates output from information in parameters at interval 0, and the only filters with information at interval 1 are devices or connected to devices. These two conditions avoid problems with filters in execution separated by delays.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Application

*Application*0

$\forall f : \mathit{filters} \bullet (\forall c : f.O \bullet ((\mathit{stream}(c)).m)(0) = \varnothing)$

$\forall d : \mathit{delays} \bullet (\forall c : d.O \bullet ((\mathit{stream}(c)).m)(0) = \varnothing)$

$\forall f : \mathit{executingFilters}(\mathit{app2Comp}(\theta \mathit{Application0}), 1) \bullet$
$f \in \mathit{devices} \vee (\exists d : \mathit{devices} \bullet (\exists p : \mathit{paths}(\mathit{filters}, d) \bullet$
$p \upharpoonright \{f\} \neq \varnothing))$

3.5.6 Changes in the Dataflow

We will consider three types of changes in the dataflow: One that changes channels in a filter, without changing the structure of its output connections, another one that changes a filter in a connection scheme by replacing it for a compatible one and connecting it to the previous scheme, and a last one when filters are removed or added to the dataflow.

Channel Changes in a Filter

We assume that a filter function can compute the new output channels from the new input ones, since we have assumed that in general we see just a subset of possible results [20]. The operation is defined in a way that each old channel keeps the same, or has only one replacement. We also define the function *isCompatible*, which says if two channels can be replaced one by the other.

$\mathit{isCompatible} : (C \times C) \longrightarrow \{0, 1\}$

$\forall c1, c2, c3 : C \bullet (\mathit{isCompatible}(c1, c1) = 1 \wedge$
$\mathit{isCompatible}(c1, c2) = \mathit{isCompatible}(c2, c1) \wedge$
$(\mathit{isCompatible}(c1, c2) = 1 \wedge \mathit{isCompatible}(c2, c3) = 1) \Rightarrow$
$\mathit{isCompatible}(c1, c3) = 1)$

For a filter, a channel change is straightforward: the new channels replace the old ones, providing that they are only pairs between these sets that are compatible.

ChangeChannels

$f : \mathit{Filter}$
$f' : \mathit{Filter}$
$\mathit{newIC}? : \mathbb{P}\, C$
$\mathit{newOC}? : \mathbb{P}\, C$
$\mathit{oldIC}? : \mathbb{P}\, C$
$\mathit{oldOC}? : \mathbb{P}\, C$

$f'.\mathit{name} = f.\mathit{name}$

$\mathit{oldIC}? \subseteq f.I \wedge \mathit{oldOC}? \subseteq f.O$

$f'.I = \mathit{newIC}? \wedge \#\mathit{newIC}? = \#\mathit{oldIC}? \wedge$
$(\forall c' : \mathit{newIC}? \bullet (\exists_1 c : \mathit{oldIC}? \bullet c' = c \vee \mathit{isCompatible}(c, c') = 1))$

$f'.O = \mathit{newOC}? \wedge \#\mathit{newOC}? = \#\mathit{oldOC}? \wedge$
$(\forall c' : \mathit{newOC}? \bullet (\exists_1 c : \mathit{oldOC}? \bullet c' = c \vee \mathit{isCompatible}(c, c') = 1))$

ChangeChannelsCF defines a change of channels in a composed filter. All references to old channels are replaced by the new ones, and the structure of the composed channel is kept, since there is only one possible replacement for each channel.

[20]See the textitBehavior schema.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ChangeChannelsCF

$$
\begin{array}{l}
\Delta ComposedFilter \\
newIC? : \mathbb{P}\ C \\
newOC? : \mathbb{P}\ C \\
oldIC? : \mathbb{P}\ C \\
oldOC? : \mathbb{P}\ C \\
\hline
name' = name \land \\
\#filters = \#filters' \land \#delays = \#delays' \land \\
\#objects = \#objects' \land \#delays = \#delays' \\
\\
\forall f : filters \bullet (\exists_1 f' : filters' \bullet \\
((oldIC? \cap f.I = \varnothing \land oldOC? \cap f.O = \varnothing \land f = f') \\
\lor ((oldIC? \cap f.I \neq \varnothing \lor oldOC? \cap f.O \neq \varnothing) \\
\land (\exists_1 ic, oc, ic', oc' : \mathbb{P}\ C \bullet ic \subseteq oldIC? \land oc \subseteq oldOC? \land \\
ic' \subseteq newIC? \land oc' \subseteq newOC? \land \\
ChangeChannels[ic/oldIC?, oc/oldOC?, ic'/newIC?, oc'/newOC?])))) \\
\\
\forall f : filters \bullet (\exists_1 f' : filters' \bullet \\
(\exists_1 c : f.O; c' : f'.O \bullet \\
(f.name = f'.name \land \{f2 : succ1(filters, f, c) \bullet f2.name\} = \\
\{f2' : succ1(filters', f', c') \bullet f2'.name\})))
\end{array}
$$

Change a Filter in a Connection Scheme

Let's now define a way to replace a filter by another in the dataflow, while connections are kept as much as possible. A filter holder allows us to replace a filter f by another, while keeping as much as possible previous connections of f. Part of the definition of a filter holder are a merge filter, a duplicator filter, and a compability function between channels similar to *isCompatible*, defined as follows:

Merge2

$$
\begin{array}{l}
Filter \\
i1 : C \\
i2 : C \\
o : C \\
\hline
I = \{i1, i2\} \land O = \{o\} \land i1 \neq i2 \land i1 \neq o \\
\\
\forall i : \mathbb{N} \bullet ((stream(o)).m)(i) = ((stream(i1)).m)(i) \uplus ((stream(i2)).m)(i)
\end{array}
$$

Duplicate

$$
\begin{array}{l}
Filter \\
i : C \\
o1 : C \\
o2 : C \\
\hline
I = \{i\} \land O = \{o1, o2\} \\
\\
\forall n : \mathbb{N} \bullet ((stream(i)).m)(n) = ((stream(o1)).m)(n) \land \\
((stream(i)).m)(n) = ((stream(o2)).m)(n)
\end{array}
$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$fhCompatible : (C \times C) \longrightarrow \{0,1\}$

$\forall\, c1, c2, c3 : C \bullet (fhCompatible(c1, c1) = 1 \wedge$
$fhCompatible(c1, c2) = fhCompatible(c2, c1) \wedge$
$(fhCompatible(c1, c2) = 1 \wedge fhCompatible(c2, c3) = 1) \Rightarrow$
$fhCompatible(c1, c3) = 1)$

$\forall\, c1, c2 : C \bullet (fhCompatible(c1, c2) = 1 \Rightarrow isCompatible(c1, c2) = 1)$

Mergers and duplicators can be seen as filters with the following functions:

$m2f : Merge2 \longrightarrow Filter$
$d2f : Duplicate \longrightarrow Filter$

$\forall\, m : Merge2; f : Filter \bullet (m2f(m) = f \Rightarrow$
$m.name = f.name \wedge m.I = f.I \wedge m.O = f.O \wedge m.fun = f.fun \wedge$
$m.delay = f.delay \wedge m.params = f.params)$

$\forall\, d : Duplicate; f : Filter \bullet (d2f(d) = f \Rightarrow$
$d.name = f.name \wedge d.I = f.I \wedge d.O = f.O \wedge d.fun = f.fun \wedge$
$d.delay = f.delay \wedge d.params = f.params)$

A *filter holder* surrounds its contained filter by a structure of filters shown in Figure 3.14. We assume the imposed structure is given by the channels with numbers and the contained filter f originally had the channels with letters.

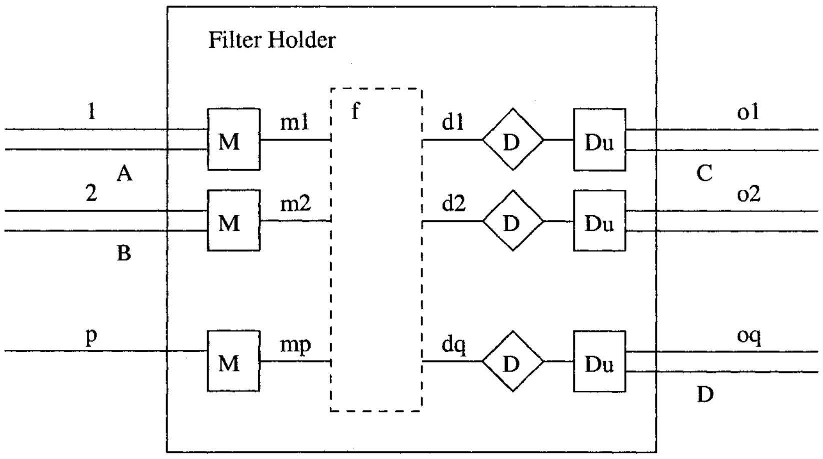


Figure 3.14: Internal Structure of an Filter Holder

The schema for a filter holder is as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{array}{l}
\textit{FilterHolder} \\
\hline
nameFH : L \\
IFH : \mathbb{P}\, C \\
OFH : \mathbb{P}\, C \\
predecessors : \mathbb{P}\, Filter \\
successors : \mathbb{P}\, Filter \\
fSet : \mathbb{P}\, Filter \\
mrgs : \mathbb{P}\, Merge2 \\
dels : \mathbb{P}\, Delay \\
dups : \mathbb{P}\, Duplicate \\
\hline
\# fSet < 2 \\
\forall p : predecessors \bullet p.O \cap IFH \neq \varnothing \\
\forall s : successors \bullet s.I \cap OFH \neq \varnothing \\
\forall ifh : IFH \bullet \exists_1 m : mrgs \bullet ifh = m.i1 \\
\forall ofh : OFH \bullet \exists_1 del : dels;\ dup : dups \bullet \\
(del.OD = dup.i \land dup.o1 = ofh) \\
fSet \neq \varnothing \land (\exists f : fSet \bullet \\
((\forall ic : f.I \bullet (\forall ifh : IFH \bullet fhCompatible(ic, ifh) = 0) \lor \\
(\exists_1 ifh : IFH \bullet fhCompatible(ic, ifh) = 1)))) \\
\hline
\end{array}
$$

Once a new filter $f2$ is assigned to an object holder, it disconnects the previous f contained, restores its original connections, and connects $f2$ inside the structure. Such a structure keeps the original connections of any filter, and avoids cycle problems, due the delays in the outputs.

$$
\begin{array}{l}
\textit{NewFilterFH} \\
\hline
\Delta FilterHolder \\
f? : Filter \\
f! : Filter \\
\hline
nameFH = nameFH' \land IFH = IFH' \land OFH = OFH' \land \\
predecessors = predecessors' \land successors = successors' \\
fSet \neq \varnothing \land (\exists_1 fOld : fSet \bullet \\
fOld.name = f!.name \land \\
f!.I = fOld.I \setminus \{mrg : mrgs \mid mrg.o \in fOld.I \bullet mrg.o\} \cup \\
\{mrg : mrgs \mid mrg.o \in fOld.I \bullet mrg.i2\} \land \\
f!.O = fOld.O \setminus \{del : dels \mid del.ID \in fOld.O \bullet del.ID\} \cup \\
\{dup : dups \mid (\exists oc : fOld.O \bullet fhCompatible(dup.o2, oc) = 1) \bullet dup.o2\}) \\
\exists f?' : fSet';\ ic, ic', oc, oc' : \mathbb{P}\, C \bullet \\
(ic = \{c : f?.I \mid (\exists_1 ifh : IFH \bullet fhCompatible(c, ifh) = 1)\} \land \\
oc = \{c : f?.O \mid (\exists_1 ofh : OFH \bullet fhCompatible(c, ofh) = 1)\} \land \\
ic' = \{m : mrgs \mid (\exists_1 ifh : IFH;\ c : f?.I \bullet \\
(fhCompatible(c, ifh) = 1) \land ifh = m.i1) \bullet m.o\} \land \\
oc' = \{del : dels \mid (\exists_1 ofh : OFH;\ dup : dups;\ c : f?.O \bullet \\
(fhCompatible(c, ofh) = 1) \land ofh = dup.o1 \land dup.i = del.OD) \bullet del.ID\} \land \\
ChangeChannels[f?/f, f?'/f', ic/oldIC?, oc/oldOC?, ic'/newIC?, oc'/newOC?]) \\
\hline
\end{array}
$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Add and Remove Filters

The addition and removal of filters — or delays — in a composed filter is straightforward: The new filter is added or removed from the set of filters [21].

$$\begin{array}{|l}
\textit{AddFilter} \\
\hline
\Delta \textit{Application} \\
\textit{newF}? : \textit{Filter} \\
\hline
\textit{name} = \textit{name}' \land \textit{delays} = \textit{delays}' \land \textit{objects} = \textit{objects}' \land \\
\textit{devices} = \textit{devices}' \\
\textit{newF}?.\textit{name} \notin \{f : \textit{filters} \bullet f.\textit{name}\} \cup \{\textit{name}\} \land \\
\textit{newF}?.O \cap \bigcup\{f : \textit{filters} \bullet f.O\} = \emptyset \land \\
(\forall o : \textit{objects} \bullet o.O \cap \textit{newF}?.I = \emptyset) \land \\
\textit{filters}' = \textit{filters} \cup \{\textit{newF}?\} \land \textit{cyclic}(\textit{filters}') = \emptyset \\
\hline
\end{array}$$

$$\begin{array}{|l}
\textit{RemoveFilter} \\
\hline
\Delta \textit{Application} \\
\textit{oldF}? : \textit{Filter} \\
\hline
\textit{name} = \textit{name}' \land \textit{delays} = \textit{delays}' \land \textit{objects} = \textit{objects}' \land \\
\textit{devices} = \textit{devices}' \\
\textit{filters}' = \textit{filters} \setminus \{\textit{oldF}?\} \\
\hline
\end{array}$$

In the same way, modifications of delays can be defined as follows:

$$\begin{array}{|l}
\textit{AddDelay} \\
\hline
\Delta \textit{Application} \\
\textit{newD}? : \textit{Delay} \\
\hline
\textit{name} = \textit{name}' \land \textit{filters} = \textit{filters}' \land \textit{objects} = \textit{objects}' \land \\
\textit{devices} = \textit{devices}' \\
\textit{newD}?.\textit{name} \notin \{d : \textit{delays} \bullet d.\textit{name}\} \cup \{\textit{name}\} \land \\
\textit{newD}?.O \cap \bigcup\{d : \textit{delays} \bullet d.O\} = \emptyset \\
\textit{delays}' = \textit{delays} \cup \{\textit{newD}?\} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\textit{RemoveDelay} \\
\hline
\Delta \textit{Application} \\
\textit{oldD}? : \textit{Delay} \\
\hline
\textit{name} = \textit{name}' \land \textit{filters} = \textit{filters}' \land \textit{objects} = \textit{objects}' \land \\
\textit{devices} = \textit{devices}' \\
\textit{delays}' = \textit{delays} \setminus \{\textit{oldD}?\} \\
\hline
\end{array}$$

3.5.7 Operations Over an Application

The operations over an application are based on the previous schemas. An *InTml application* is a well defined application with object holders:

[21] It is important to notice that these two operations have to fulfill all previous requirements for composed filters; in particular, cycles without delays are not allowed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{array}{l}
\textit{InTmlApp} \\
\hline
\textit{Application} \\
\textit{fhs} : \mathbb{P}\, \textit{FilterHolder} \\
\hline
\forall \textit{fh} : \textit{fhs} \bullet \textit{fh.IFH} \subseteq \textit{IU} \land \textit{fh.OFH} \subseteq \textit{OU} \\
\land\ \textit{fh.predecessors} \subset \textit{filters} \\
\land\ \textit{fh.successors} \subset \textit{filters} \\
\land\ \textit{fh.fSet} \subset \textit{filters} \land \textit{fh.dels} \subseteq \textit{delays} \\
\land\ \{\textit{mrg} : \textit{fh.mrgs} \bullet \textit{m2f}(\textit{mrg})\} \subset \textit{filters} \\
\land\ \{\textit{dup} : \textit{fh.dups} \bullet \textit{d2f}(\textit{dup})\} \subset \textit{filters} \\
\hline
\end{array}
$$

The execution of an InTml application is either the normal flow of messages, or one of the following special changes: channel changes, object holder executions, addition of filters, removal of filters, addition of delays, and removal of delays. The normal dataflow execution is similar to the execution of any filter:

$$
\begin{array}{l}
\textit{InTmlExecuteDataflow} \\
\hline
\Xi \textit{InTmlApp} \\
\textit{inputD?} : \mathbb{P}(\text{bag}\, M) \\
\textit{output!} : \mathbb{P}(\text{bag}\, M) \\
\textit{i?} : \mathbb{N} \\
\hline
\forall \textit{in} : \textit{inputD?} \bullet (\exists\, c : I \bullet \textit{in} = ((\textit{stream}(c)).m)(\textit{i?})) \\
\forall \textit{out} : \textit{output!} \bullet (\exists\, c : O' \bullet \textit{out} = ((\textit{stream}(c)).m)(\textit{i?})) \\
\hline
\end{array}
$$

A change of channels is defined on top of the changes in a composed filter, with extra conditions for filter holders that assures that if the change involves a filter connected to a filter holder, the change will keep the structure of the dataflow as it was before:

$$
\begin{array}{l}
\textit{InTmlExecuteChangeChannels} \\
\hline
\Delta \textit{InTmlApp} \\
\textit{newIC?} : \mathbb{P}\, C \\
\textit{newOC?} : \mathbb{P}\, C \\
\textit{oldIC?} : \mathbb{P}\, C \\
\textit{oldOC?} : \mathbb{P}\, C \\
\hline
\textit{ChangeChannelsCF} \\
\forall \textit{fh} : \textit{fhs} \bullet (\exists_1 \textit{fh}' : \textit{fhs}' \bullet \\
(\textit{fh.IFH} \cap \textit{newIC?} = \varnothing \land \textit{fh.OFH} \cap \textit{newOC?} = \varnothing \land \\
\textit{fh} = \textit{fh}') \lor \\
((\textit{fh.IFH} \cap \textit{newIC?} \neq \varnothing \lor \textit{fh.OFH} \cap \textit{newOC?} \neq \varnothing) \\
\land (\textit{fh.nameFH} = \textit{fh}'.\textit{nameFH} \land \textit{fh.predecessors} = \textit{fh}'.\textit{predecessors} \land \\
\textit{fh.successors} = \textit{fh}'.\textit{successors} \land \textit{fh.fSet} = \textit{fh}'.\textit{fSet} \land \\
\textit{fh}'.\textit{IFH} = \textit{fh.IFH} \setminus \textit{oldIC?} \cup \{m : \textit{fh}'.\textit{mrgs} \bullet m.\textit{i1}\} \land \\
\textit{fh}'.\textit{OFH} = \textit{fh.OFH} \setminus \textit{oldOC?} \cup \{d : \textit{fh}'.\textit{dups} \bullet d.\textit{o1}\}))) \\
\hline
\end{array}
$$

The execution of filter holders in an InTml application is based on the previous schema *NewFilterFH*, with an additional condition to assure that nothing else changes:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

InTmlExecuteFHs

$\Delta InTmlApp$
$f? : Filter$
$f! : Filter$

$f? \in filters \wedge f! \in filters$

$name = name' \wedge delays = delays' \wedge objects = objects' \wedge$
$devices = devices'$

$\exists_1 fh : fhs; fh' : fhs'; NewFilterFH \bullet$
$(fh.nameFH = nameFH \wedge fh.IFH = IFH \wedge fh.OFH = OFH \wedge$
$fh.predecessors = predecessors \wedge fh.successors = successors \wedge$
$fh.fSet = fSet \wedge fh.mrgs = mrgs \wedge fh.dels = dels \wedge$
$fh.dups = dups \wedge fh'.nameFH = nameFH' \wedge fh'.IFH = IFH' \wedge$
$fh'.OFH = OFH' \wedge fh'.predecessors = predecessors' \wedge$
$fh'.successors = successors' \wedge$
$fh'.fSet = fSet' \wedge fh'.mrgs = mrgs' \wedge fh'.dels = dels' \wedge$
$fh'.dups = dups')$

Adding and removing filters are directly defined over the previous schemas *AddFilter* and *RemoveFilter*, respectively:

InTmlExecuteAddFilter

$\Delta InTmlApp$
$newF? : Filter$

$fhs = fhs'$

$AddFilter$

InTmlExecuteRemoveFilter

$\Delta InTmlApp$
$oldF? : Filter$

$fhs = fhs'$

$RemoveFilter$

Similarly changes in delays are defined as follows:

InTmlExecuteAddDelay

$\Delta InTmlApp$
$newD? : Delay$

$fhs = fhs'$

$AddDelay$

InTmlExecuteRemoveDelay

$\Delta InTmlApp$
$oldD? : Delay$

$fhs = fhs'$

$RemoveDelay$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Finally, an *InTmlStep* is the execution of any of the tasks that change the state of the dataflow, followed by the execution of the dataflow.

$\mathit{InTmlStep} \mathrel{\widehat{=}} (\mathit{InTmlExecuteChangeChannels} \vee \mathit{InTmlExecuteFHs} \vee$
$\mathit{InTmlExecuteAddFilter} \vee \mathit{InTmlExecuteRemoveFilter} \vee$
$\mathit{InTmlExecuteAddDelay} \vee \mathit{InTmlExecuteRemoveDelay}) \wedge$
$\mathit{InTmlExecuteDataflow}$

3.6 Properties of this Architecture

This architecture has the following features:

- Filters can run in different processors. There are no restrictions on the simultaneous execution of filters, and how they send information to followers. This allows parallel implementations and also sequential ones, with the same semantics.
- A filter "knows" all events that are originated in the same time frame and received through its input channels. This allows specific implementations to filter unnecessary information.
- The state of the world, reflected in terms of the state of the objects in the application, is consistent for all filters during the execution of a time frame. This avoids side effects, as in other dataflow–based implementations such as VRML, related to the order of execution of filters.
- Composed filters have clear recursive semantics and allow complexity management, by hiding unnecessary details.
- An application can have as many devices as required, all treated in a uniform way.
- The particular implementation of the behavior of a filter is hidden from the dataflow point of view. In this way, we can separate the high level design of the dataflow and the low level design of behaviors and interaction techniques.
- Object holders define a mechanism similar to pointers in common programming languages, and they are very useful for the definition of dynamic changes.
- The semantics described here can be implemented in several platforms from a simple desktop computer to a massively parallel computer. In this way, a VR application is scalable to a wide variety of hardware platforms, while it keeps the same semantics.
- A designer can easily identify which filters in a dataflow are specific to a particular hardware platform. This allows designers to perform a process of reaccomodation of VR applications to different hardware platforms and interaction styles. We call such a process retargeting, and it will be part of the methodology we define later in this thesis.

There are also some issues that require further research, such as:

- Actual comparative results between a parallel and a sequential implementation of InTml have to be performed. Ideally, we should be able to create some analytical methods for measuring performance advantages in a parallel implementation of a specific application, but we have to explore this possibility in more in detail.
- InTml is a new component technology that takes into account the specific quality attributes required in VR applications. However, we require more applications implemented in this technology to validate more thoroughly its advantages and features. Such experimental tests will require a community of users and an active developer community to support them.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Exact time management and synchronization are issues not directly visible in InTml. This allows designers to worry about requirements without taking care of synchronization issues. However, such issues should be addressed in a succesful VR application. Our current approach forces designers to hide synchronization issues under filters that transparently handle several streams into a synchronized one. This approach has to be tested in more detail to understand its implications.

- More development tools are necessary in order to provide a true professional environment for VR designers.

- As InTml libraries grow, tools for finding and organizing them will be required in order to make them usable and avoid work repetition.

3.6.1 An Example of InTml Interpretation

This specification allows us to understand the semantics behind an InTml application representation. For example, if we analyze again Figure 3.2 one can infer the following characteristics:

- All filters in the application can be executed in one step, since all filters are reachable from the device `handTracker` without delays in the middle [22]. The `handTracker` sends changes to both the object and the selection technique. The selection technique takes the new coordinates and the current object state and decides if the object will collide any other object in the scene, once the new coordinates are set. If this is the case, the collided object is sent to the feedback technique. All these operations are executed at the same interval. Once the dataflow execution has finalized, changes in objects are executed (In this case, new position and orientation for `handRepr` from the tracker).

- Since *InTmlExecuteFHs* is executed before *InTmlExecuteDataflow*, the object `handRepr` is assigned to the object holder `handRepresentation` before events from the tracker arrive (i.e., `position` and `orientation`).

- Another representation of the same application is shown in Figure 3.15, with a slightly different meaning but with the same results. In this case, changes in position and orientation are propagated after they are executed in the object. This representation executes the entire dataflow in two steps: the first one with {`handTracker, handRepresentation`} and the second with {`SelectByTouching, Feedback`}. This approach might be preferred if the object can decide if it can do the requested operation. In this case, the processing is executed first, and if the object decides that the change can be done, it will propagate the changes to the selection technique. Also note that when the selection technique executes, `handRepr` actually has the same position and orientation as the ones received in the input ports `position` and `orientation`. We allow then `SelectByTouching` to operate in two slightly different modes, since it does not take into account the actual position and orientation of the object.

- The actual description of the function executed by a filter is hidden from the diagram, and in the current InTml implementation, it is defined in a textual description attached to each filter class. For example, the function `SelectByTouching` is described as follows: It is a selection technique that takes an object and checks if it will collide with another object in a scene after moving to a new position and orientation, received as parameters. Addition and removals of objects in the scene are allowed, but those changes have to be explicitly informed. It is possible to design this selection technique in a different way, and Figure 3.16 shows a different one, with the following interpretation: it is a selection technique that takes the current position of an object and checks if it collides with another object in the scene. Changes in the scene or in the

[22] See definition of *executingFilters* in Section 3.5.5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

object are implicitly taken into account. This definition might be preferred, since it allows less modes of operation and avoids possible misinterpretations of its execution. It might also be defined in terms of the previous one. However, it takes two intervals to be totally executed, due the inner object holders.

- If the first representation of `SelectByTouching` is used, `handRepr` will receive two copies of the events from `handTracker`: one from the object holder inside the selection technique, one from `handRepresentation`. While this might be redundant it does not affect the semantics of the application in any way, and compiler techniques can be used to avoid this redundancy.

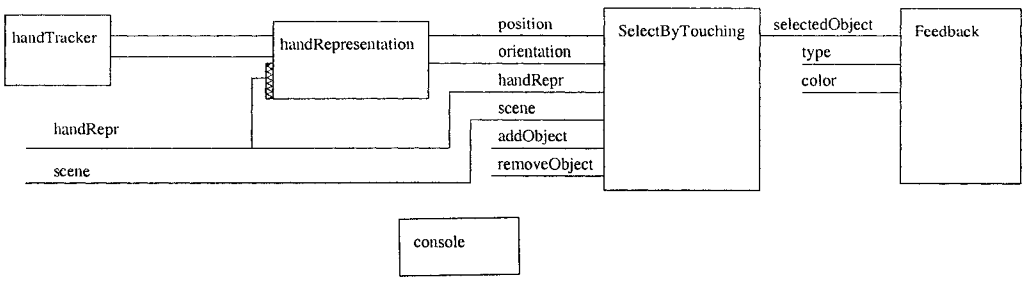


Figure 3.15: A Modified Version of a Simple Application.

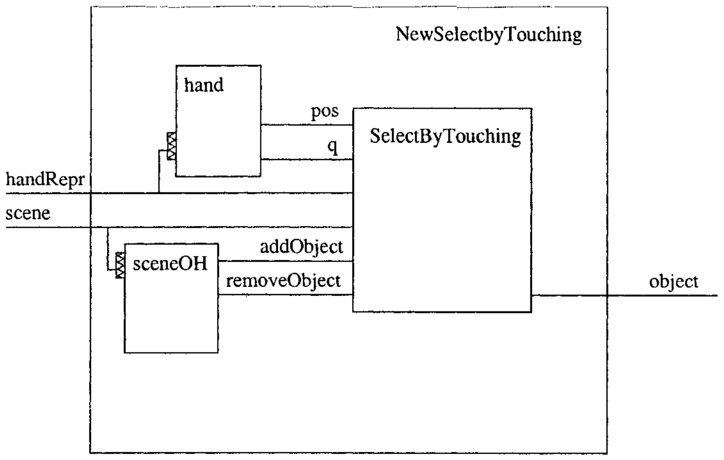


Figure 3.16: A Modified Version of SelectByTouching.

3.6.2 Lessons Learned from the Z Language Description

Our attempt to formally describe the semantics of InTml gave us a better understanding of what our model is, and what are its capabilities. We started our first implementation after prototyping three environments for VR development, which gave us a good understanding of the requirements of such an endeavour. However, the description in the Z language that we started after the first implementation gave us a more generic, clean, and scalable description than the one that we implemented. We extended the concept of object holders to allow any type of filter, which makes InTml a second–order language (filters can have filters as arguments).

Formalization allowed us to better understand the meaning of an InTml application than from just a drawing, since the inner semantics of each element and the overall execution

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

model is clearer. The formalism allowed us to cleanly separate the meaning of a filter from the concept of a filter holder, and what could be the type of a holder. It also allowed us to make clearer the differences and similarities between applications and composed filters.

The concept of a delay emerged from the work Lee and Parks [57] as a very useful addition to the set of concepts in InTml. Despite the fact that it does not yet appear in the XML syntax, a delay is a very useful abstract concept that allows us to understand the execution model, how cycles are executed, and how object holders work.

An important concept in the Z description is the one of dynamic changes in the structure of the dataflow, that will allow us in the future to offer a richer language, i.e. with support for adding or deleting filters, changes in connections, delays at the XML level, and filter holders.

Finally, the description in the Z language also allowed us to compare the semantics of our dataflow proposal with other dataflow proposals, and reuse part of their notation and their semantics. The differences of our proposal with previous ones in the area of dataflow based computation are clearer, and new questions have emerged for future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Development Process based on InTml

In this chapter, we describe an InTml–based development process for 3D interactive applications. The process establishes a collaboration between two types of people with complementary skills and different viewpoints. We show how our test application was developed, how our methodology differs from other methodologies for 3D applications. We also show some metrics about development of multi-platform VR applications from the CVS repository of our example application.

4.1 Process Description

Our development process divides tasks between two groups of people: designers and developers. Designers are in charge of the overall design of the application. They know about InTml, its semantics, and the components in the InTml library. Developers are in charge of the fine–grain details inside components, and know how components can be implemented on top of available frameworks and libraries. Each role develops complementary tasks: designers are closer to end–users of the application, while developers are closer to the programming and hardware details of the solution.

The first version of a VR application, targeted to a particular hardware platform, is created by pursuing the tasks in Figure 4.1. Such a process guarantees a clear division between the architecture of the solution and the implementation of each component in the architecture. Once a first version is completed, new versions can be created by retargeting this application to other hardware platforms. Figure 4.2 shows the process for retargeting an existing VR application, with general references to the processes in Figure 4.1. Collaboration in Figure 4.2 refers to the relationships between designer and developer tasks in Figure 4.1.

There are other methodologies for the development of VR applications, such as the one by Tanriverdi and Jacob [93]. While such alternatives have similarities with the one presented here, ours extend them with the concept of retargeting and the concept of separation of roles. Further comparison can be found in Section 4.4.

Content components, such as geometric models for objects, special graphic effects, sound, or haptics, are designed with the aid of third party tools. It is necessary that all created media types can be understood by the foundation framework where the InTml application will run.

Since InTml can be implemented on top of several foundation frameworks, it is possible to discover platform limitations in the process of developing new applications. Such limitations can be detected by developers while trying to create new components. In this case developers and designers can compromise in a solution that both satisfies requirements and minimizes changes in the foundation framework. However, the more mature an InTml implementation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is, the less these changes will be required.

The following sections describe the tasks in our methodology in more detail.

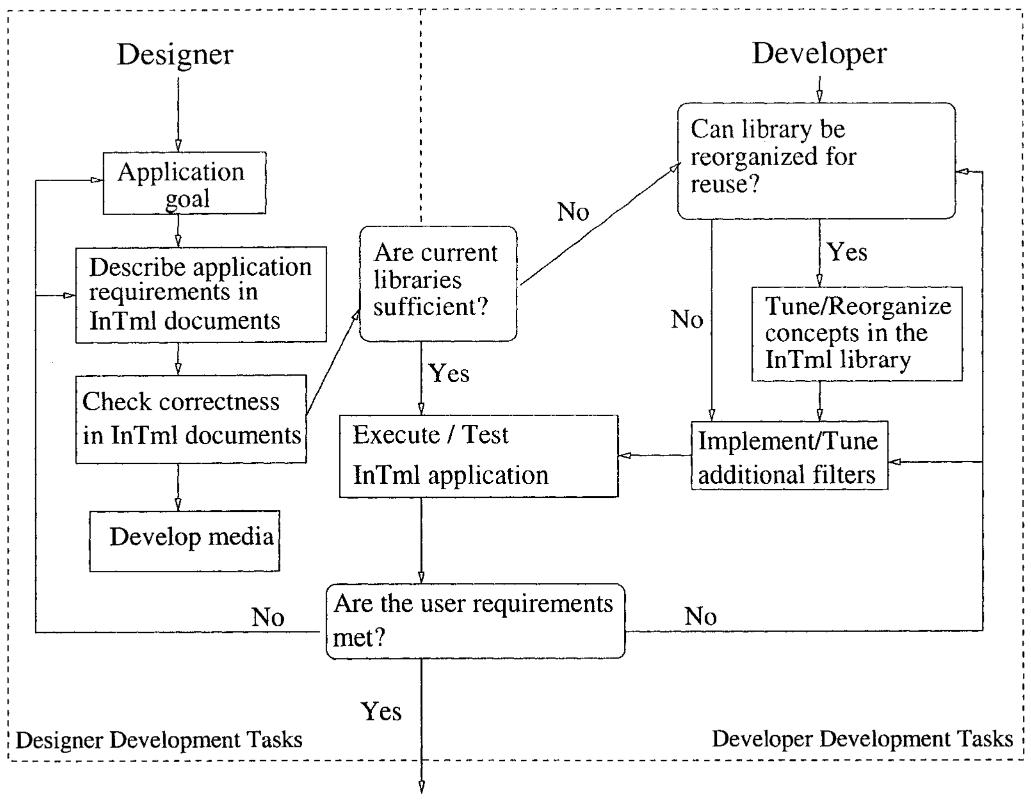

Designer
Developer
Application goal
Describe application requirements in InTml documents
Check correctness in InTml documents
Develop media
Are current libraries sufficient?
No
Yes
Can library be reorganized for reuse?
Yes
No
Tune/Reorganize concepts in the InTml library
Implement/Tune additional filters
Execute / Test InTml application
Are the user requirements met?
No
No
Yes
Designer Development Tasks
Developer Development Tasks

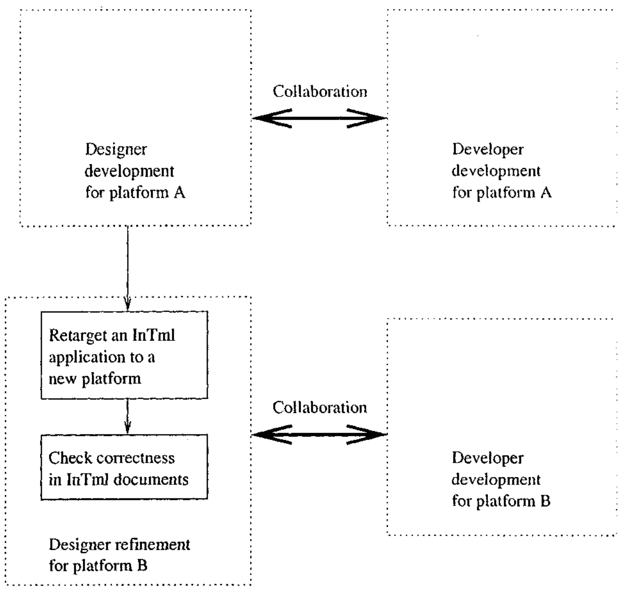
Figure 4.1: Collaborative development process for a particular hardware and software platform.

4.1.1 Application Goal

This is the task of eliciting user's requirements. Designers identify main tasks in the application, and main InTml components in the solution. Techniques borrowed from requirement gathering in traditional development methodologies, such as the ones in Kulak and Guiney [55] or Mayhew [62], can be applied here.

The result of this task is a description of user requirements, the tasks to be implemented in the VR application, the quality attributes of the desired implementation, and an understanding of the hardware platform to be used.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Collaboration
Designer development for platform A
Developer development for platform A
Retarget an InTml application to a new platform
Check correctness in InTml documents
Designer refinement for platform B
Collaboration
Developer development for platform B

Figure 4.2: Adapting a VR application to a new hardware and software platform.

4.1.2 Description and Refinement of Application Requirements in InTml Documents

With information about user's requirements and desired quality attributes, designers work on the required InTml components and relationships. The mapping between requirements and InTml concepts can be influenced by usability rules and techniques described elsewhere [85, 88].

Designers should balance several forces in this task. Components may or may not be dependant on the hardware platform on which they will run, or the specific application domain. Platform–dependent components are easier and faster to describe, but usually more difficult to reuse in other applications. In the same way, application– dependant components can be specifically tailored to solve user requirements, but they have less value for a general–purpose library. Since our objective is to facilitate the process of application retargeting, we encourage designers to develop platform–independent components, since they can be more easily reused in other environments. They could be application–dependant, since these types of components do not affect the ability of retargeting.

An application can be divided in several files, some of them with reusable classes, some with generic descriptions of applications, and some with the actual application as it will be deployed to the users. Such a division facilitates future reuse and support for other applications in a particular platform.

4.1.3 Check Correctness in InTml Documents

InTml documents can be validated in two ways: by checking XML syntactic rules and by revising its semantics. There are several XML tools to validate document syntax, by analyzing the document's conformance to its schema [98, 27, 38]. This validation assures that documents can be parsed by XML tools, but this does not guarantee that the semantic of the document is correct. For a more thorough checking, InTml documents can be validated against the InTml semantics in the Z specification language, presented in Section 3.5. Such a process is similar to a design peer review, with the difference that the reference semantic model is described in Z. However, this task is not necessary in most cases, since it is equivalent to a process of revising an application against its functional semantics. For example, it

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is equivalent to check a VRML program against the VRML specification. This task is not performed everytime a new application is created, and usually the execution of a program is enough to understand its meaning. Designers with a general knowledge of InTml, given by an understanding of the more informal descriptions in Section 3, should be able to develop medium size VR applications [1].

Section 6 will present a simple tool for checking InTml documents. This is an intermediate step between basic XML syntax checking and a peer review against the formal description. We think this will be the more efficient way to check InTml documents, once the checker has enough functionality.

4.1.4 Completeness Test for Current Library

Designers create new applications by reusing filters in the InTml library [2] and by defining new filter classes, to be created by developers. In this stage designers check if the available components in the library are enough to describe the new application. If not, designers create new InTml classes and ask developers to implement them.

Developers can also decide to improve the coverage of the library by adding new application–independent components. This is a way to improve the library, but since we concentrate here on retargeting, we will not describe this task in more detail.

4.1.5 Structure and Reusability Test for Current Library

Once designers request new components by defining new InTml classes, developers should create the code behind them. At this moment they should consider to reorganize the current implementation scheme of the library, since it may be worth to reuse parts of other classes already in the library.

4.1.6 Implementing or Tuning of Additional Filters

InTml can be implemented on top of several foundation frameworks or libraries, such as Performer, Java3D, VRJuggler, VRPN, and so on. Once an InTml application has been defined, developers should translate this representation to the core frameworks. Ideally, code generators should be available to aid in such a process, in order to keep general restrictions and characteristics of the underlying implementation. In this way, the work of a developer consists of generating "templates" of code for new components, and filling the blanks for the particular behavior of such components.

A developer has to take into account the semantics of InTml components when doing an implementation. In particular, filters can not directly change the scene graph, and objects can only apply changes for the next interval in time [3].

Developers should integrate new code to the InTml runtime environment, so designers can use the new filters and test them in InTml applications. A good practice is to create a simple test application for any new filter.

4.1.7 Tuning or Reorganizing of Concepts in the InTml Library

If new filter classes are requested, developers should create such new classes either from scratch, or by reusing other implemented classes. The purpose of this task is to reorganize class implementations when new classes can be integrated in a better way to the rest of the library. The relationships in implementation does not necessarily have to be reflected at the InTml code, i.e. it is possible to reuse code of a class that is not related in InTml.

[1] In the same way that VRML programs can be developed without understanding the fine details of its specification.

[2] The InTml library is defined in Section 6.3.

[3] Our current implementation fulfills this condition by executing filters before any object at any time frame.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This task assures an organized evolution of the InTml library implementation, since code reorganization is considered when new classes are required.

4.1.8 Executing and Testing of a New InTml Application

Designers and developers test the new classes and the new application that uses them. Tests can be done separate for each class, or as a subset of the application. Developers should care about the performance of the implementation and overall quality of the algorithms in the solution. Designers should care about conformance to the given specification in InTml, and other possible uses, i.e. when not all input ports are connected.

4.1.9 Test Coverage of User Requirements

Designers check if the current application covers user requirements stated before. If users do not agree about the coverage, or if new requirements should be included, designers can start a new cycle over these development tasks. Iteration over the previous tasks also allows for evolutionary development of a VR application.

4.1.10 Retargeting of An InTml Application to A New Platform

This task retargets an existing VR application to a new platform. The process involves changes in devices, interaction techniques, and content elements, according to the availability and common practices in the new platform. Our purpose is to change the mechanisms used to implement the application tasks in the current platform to the more suitable ones in the new platform, while keeping the overall application functionality similar.

This adaptation process includes the following steps:

- Change of devices to the ones available in platform `B`.
- Creation of adapters in order to simulate the type of information that was received from devices in platform `A`. For example, if we move from a tracker-based system to a mouse and keyboard based system, an adapter can be created to simulate the tracker output by mouse and keyboard events. This is the process of porting an application to new hardware, without changing interaction techniques.
- Replacement of behaviors, interaction techniques, and widgets, for the ones that are more suitable in platform `B`. For example, if the selection technique in platform `A` is based on collision with a virtual hand, we can decide to change this in platform `B` to selection by intersection with a ray. This replacement might propagate changes to the entire application.
- Addition or removal of tasks, behaviors, interaction techniques, and widgets that are platform–dependent. For example, the coordination of movement of the image and the head in a HMD based environment should be removed in other platforms.

Developers repeat the development process in platform `B`, reusing code or designs from platform `A` when possible, and interacting with designers until user requirements have been met. In this way, the application keeps the same functionality in both platforms, while also taking into account the particular advantages of each one of them. More iterations of the entire process can create evolved versions of the application in each platform, with improved functionality.

Development time for new versions is reduced for two reasons: Components from previous versions of the application can be reused in new applications, and components in the InTml library can also be reused. Since InTml components are self–contained and independent from the environment, reusing them is straightforward. Also, since the semantics of InTml helps designers to understand what a component does without knowing the implementation details, the apparent complexity of the application is reduced.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2 Example: A Matching Application

As an example, we describe here our experience with an application to measure performance for tasks such as selection, translation, and rotation of 3D objects. This application was developed as a proof of concept of the retargeting process, and also as a study of user performance in different hardware setups [4]. The application shows three objects in the user's visible area, with positions and orientations chosen at random. Three replicas are shown in different positions and orientations, and the user's goal is to select, move, and rotate the objects to match their replicas.

We decided to test retargeting in different hardware platforms for these tasks. The hardware platforms that we have available are:

- A standard PC environment (standard-PC).
- An environment with a head–mounted display (HMD) and a joystick (HMDJ).
- A commercially available SMART Board [5].
- A PC environment with a Space Mouse (3D Desktop) [6].
- The Visroom, a CAVE-like environment with three stereo projectors [7].

The standard-PC, HMDJ, SMART Board, and 3D Desktop platforms use Microsoft Windows and Java3D as the foundation framework, with some specific drivers in the case of the latter two, while the Visroom uses IRIX, Performer and VRPN. Since the input and output devices of these hardware platforms have different characteristics, interaction techniques have to accomodate and use the available resources. Users will be able to execute the same tasks in all platforms, but in a way that devices afford.

The following sections describe some details for each task in the development process.

4.2.1 Application Goal

The following are the application tasks we are interested in. We give here an informal description, since the application is very simple.

- Load objects' geometry. Java3D and VR-Juggler allow us to load objects in different formats, such as 3D Max or OBJ, so we use files in these formats.
- Localize and orient objects at random.
- Create copies of objects and localize them at random.
- Define the set of objects that will be selectable (the original ones) and the ones which are not (the replicas).
- Give user's feedback when an object is selected. We change the object's color to green.
- Grab and release an object. These actions are usually joint with the rotation and translation of an object.
- Rotate and translate an object.
- Compute matching function. This function defines when an object and its replica are close enough, in position and in orientation to be considered "matched".

[4]We will discuss the user study later in this document

[5]SMART Board is a trademark of SMART Technologies Inc.

[6]Space Mouse is a trademark of 3Dconnexion

[7]Although the design for this platform is already described in InTml, its complete implementation is not finished yet.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Delete objects once they match. We decided to eliminate objects once they match in order to provide feedback.
- Log start and stop times, the user's identifier, and other important events of the experiment: user's movements, information about the objects' selection, movement, orientation, and matching.
- Finish the application.

4.2.2 Description and Refinement of Application Requirements in InTml Documents

The entire InTml files for these application are shown in Appendix A, and we show here some excerpts. Some of these files are defined by the developers according to the available functionality in the foundation frameworks, and some by the designers, according to the particular requirements of the users.

The files defined by developers are:

- library.intml: Initial set of filters, objects, and devices. Examples of elements defined here are the standard input devices — mouse, keyboard, joystick —, 3DOF and 6DOF trackers, available selection techniques, and an object in the scene.
- genericPC.intml: Generic functionality for any PC–based application.
- genericHMDJ.intml: Generic functionality for any HMDJ–based application.
- genericSMARTBoard.intml: Base application in the SMART Board.
- genericVisroom.intml: Base application in the Visroom.

For example, Listing 12 shows the description of a generic HMDJ application as the definition of devices in such a platform (keyboard, joystick, tracker, and hmd in lines 5–11), the expected objects (the current viewpoint and the rendered scene in lines 12–13), and the standard connection between the orientation of the head and the orientation of the viewpoint (lines 15–16). `id` stands for identifier, `iE` for input element, `iP` for input port, `oE` for outputElement, and `oP` for output port [8].

The files defined by designers are the following:

- newClasses.intml: New types of filters defined by the designers and used in the applications. For example, there are filters for moving objects to random positions, for replicating objects, for rotation and translation of objects, for computing the matching function.
- matching-pc.intml: The matching application in the PC environment.
- matching-hmdj.intml: The matching application in the HMDJ environment.
- matching-smartboard.intml: Matching application, SMART Board based
- matching-visroom.intml: Matching application, Visroom based.

For example, Listing 13 shows the InTml code for selection in a PC environment. A selection filter (`selection`, in lines 4–5) two objects (`obj1`, `obj2` in lines 6–9) and a scene (`selectableObjs` in line 10) are created. Objects are included in the scene by sending them to the `addObject` port of the scene (lines 11–14) [9]. Finally, the selection technique is connected to its input, the position of the selection ray in the image plane `mousePos`, and the current set of selectable objects (lines 15–19).

[8]Filters for reduction of noise from the tracker might be necessary, but they are not described here for simplicity.

[9]Currently, the InTml syntax uses `Binding` to simulate `Send`.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 12 Generic HMDJ application. HMDJ Code

```
<App id="matchingTest.genericHMDJApp">
  <ShortDesc>Generic app for an
      I-glasses HMD and a Joystick
  </ShortDesc>
  <Import id="matchingTest"/>
  <IDevice id="keyboard"
        type="GenericKeyboard"/>
  <IDevice id="joystick"
        type="GenericJoystick"/>
  <IDevice id="tracker"
        type="Generic3DOFTracker"/>
  <ODevice id="hmd" type="IGlasses"/>
  <ObjectHolder
    id="theCurrentViewpoint"/>
  <ObjectHolder
    id="theRenderedScene"/>

  <Binding iE="tracker" iP="q"
    oE="theCurrentViewpoint"
    oP="setQ"/>
</App>
```

Listing 13 Selection in the PC environment. InTml code

```
<App id="matchingTest.matchingAppPC">
    ...
  <!-- selectableObjs selectable -->
  <Filter id="selection"
        type="SelectByRay"/>
  <Object id="obj1"
        filename="file1.3ds"
        type="VRObject"/>
  <Object id="selectableObjs"
        type="Scene"/>
  <Send info="obj1"
        oE="selectableObjs"
        oP="addObject"/>
  <Binding iE="mouse"
        iP="mousePos"
        oE="selection" oP="pos"/>
  <Send info="selectableObjs"
        oE="selection" oP="scene"/>
    ...
</App>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.3 Check Correctness in InTml Documents

We checked our documents with the stand–alone application described in section 6.4.3. The tool checks the existence of filter types, type correspondences, and name validity. The use of this tool allows designers to detect problems in the InTml description before its compilation.

4.2.4 Completeness Test for the Current Library

During development, designers evaluated the completeness and relevance of the InTml library [10] for the matching application. In general, device descriptions were found in the library, whereas application dependant behavioral components were added [11]. It is possible to extract some of the filter classes in the application–dependent file to the general library, such as interaction techniques for rotation and ways to create transparent copies of objects. However, work concentrated on retargeting, not building the library, and this task was left out of the scope of this development.

4.2.5 Implementing or Tuning of Additional Filters

Developers implemented new filter classes for the application. The process was aided by a code generator, that takes InTml class descriptions and transform them into Java code [12]. Once the initial version of the Java code is generated, developers complete the required code for the general behavior of the new filter class. In our implementation, a developer had to complete three pieces of code: how to read parameters from a text file, how to process input data before execution, and how to execute in a certain frame.

Once code for a new filter class was compiled and tested, it was added to the runtime environment, by adding a line in a configuration file that specifies the file name for the object code of a particular InTml class, identified by its full name (package name and class name).

Many parameters have to be defined in the implementation, sometimes with negotiation between designers and developers, since the InTml description defines only how components can be connected to others in an application. We consider such details to be generally out of the scope of what a designer needs to know, and more related to the job and skills of developers. For example, developers should define the near and far clipping planes for the view, taking into account the requirements of this application. However, developers should be able to express technical limitations to designers, in order to achieve the best performance of a particular implementation.

4.2.6 Tuning or Reorganizing of Concepts in the InTml Library

Developers created the code in Java for all InTml classes defined by designers. During development, several components reused previous classes due to similarities in the task they developed. Some examples are the following:

- Classes for system management were organized in an inheritance hierarchy. Such components are implicit in each platform's InTml execution environment.
- The way we orient and localize objects at random changed during development. The new class reused code from the previous implementation.
- The way we create copies of objects changed during development. The new alternative is implemented on top of the previous one, by adding a parameter for the required changes.

[10]The InTml library is contained in the file *library.intml*.

[11]Application–dependent classes are defined in the file *newClasses.intml*.

[12]The code generator is described in Section 6.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.7 Executing and Testing of a New InTml Application

Once components are ready to be integrated in an application, we tested them together. Our approach was to incrementally test components and tasks in the application, so we created several applications with partial functionality. Figure 4.3 shows the separation of tasks we had in the seven testing applications we created, from the overall set of requirements previously given.

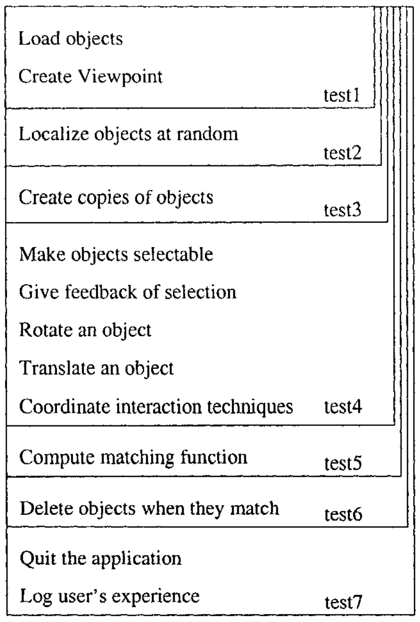


Figure 4.3: Tasks covered by several testing applications.

4.2.8 Test Coverage of User Requirements

We evaluated the coverage of user requirements once the application was in testing. This evaluation resulted in extra functionality, i.e. extra log functionality to save user and platform identifications. In this case there was no need for extra filter classes, just extra ports in already existing ones.

4.2.9 Retarget An InTml Application to A New Platform

We produce a new InTml file for the matching application in other platforms by changing devices, interaction techniques, and objects in the PC version. Generic tasks stay the same, such as how to find positions for objects, or how to decide when two objects match. However, some tasks change, since different input and output devices afford different interaction techniques. For example, we show the code for the selection interaction technique in both the HMDJ and the Visroom platforms. Listing 14 shows the selection technique in the HMDJ platform, which is based on a full 3D ray defined by the user's eye position, the joystick position, and the current center of the field of view (eyePos, in lines 8–9) [13]. Listing 15 shows the selection technique code for the Visroom platform. In this case we use the Go-Go selection technique by Poupyrev [71], and we connect it to the hand representation (lines 6–7, 16), and the position and orientation of both head and hand trackers (lines 8–15). Despite the fact the task is the same, the type of information expected for the selection technique and the type of information gathered by devices are totally different.

[13]The computation for the 3D ray is done inside HMDJRay.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Nevertheless the application has many similarities with the one for a PC environment, so much of the code in them is the same. Note that different implementations of the same task can be compared at the InTml level by designers. We consider this is one of the key advantages provided by our methodology.

Listing 14 Selection in the HMDJ environment. InTml code

```
<App id="matchingTest.mAppHMDJ">
    ...
  <!-- selectableObjs selectable -->
  <Filter id="selection"
        type="SelectBy3DRay"/>
  <Filter id="hmdj2Ray"
        type="HMDJ2Ray"/>
  <Constant id="eyePos"
        value="0 0 0"/>
  <Send info="eyePos"
        oE="hmdj2Ray" oP="headPos"/>
  <Binding iE="tracker" iP="q"
        oE="hmdj2Ray" oP="headQ"/>
  <Binding iE="joystick" iP="pos"
        oE="hmdj2Ray" oP="jPos"/>
  <Binding iE="hmdj2Ray" iP="posRay"
        oE="selection" oP="pos"/>
  <Binding iE="hmdj2Ray" iP="qRay"
        oE="selection" oP="q"/>
  <Send info="selectableObjs"
        oE="selection" oP="scene"/>
    ...
</App>
```

4.3 Some Metrics

Although we do not have information about time spent in each task during the development of our example application, we can extract some information from the CVS repository used by developers. In general, the development of the first four versions mentioned in Section 4.2 took partial time during 6 months for two developers, with some interruptions during that time. At the end of this process the applications were extensively used for a user study that is described in Section 5.2.

One way to describe the amount of work per application version is to see changes in files registered in our CVS–based repository (addition and removal of lines). Although this information is limited since the initial file size is not recorded in the CVS reporting tool and some classes are more important than others in the application, such changes give a good idea of the amount of effort dedicated per hardware platform. Figure 4.4 shows changes in number of lines of code during time, in files grouped by hardware platforms (*All* refers to files that were common to all versions). Each point in the graph of each platform is computed as the addition of all insertions plus deletions in all files related to such platform [14].

[14] A file was classified in only one category of the 5 options.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 15 Selection in the Visroom environment. InTml code

```
<App id="matchingTest.mAppVisroom">
    ...
    <!-- selectableObjs selectable -->
    <Filter id="selection"
          type="GoGoSelection"/>
    <Object id="handRepr"
          filename="hand.3ds"
          type="VRObject"/>
    <Binding iE="head" iP="pos"
          oE="selection"
          oP="posHead"/>
    <Binding iE="head" iP="q"
          oE="selection"
          oP="qHead"/>
    <Send info="handRepr"
          oE="selection"
          oP="handRepr"/>
    <Send info="handRepr"
          oE="visibleObjs"
          oP="addObject"/>
    ...
</App>
```

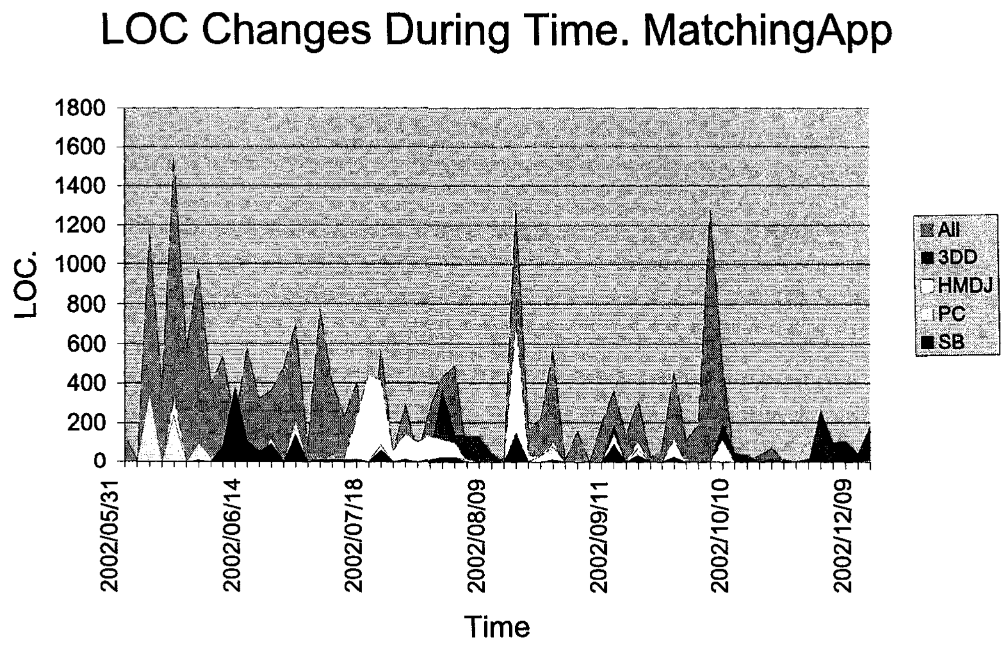


Figure 4.4: Changes in Lines of Code per Hardware Platform.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We can notice that general files dominate changes during time which means that most of the effort in development can be shared among several versions of the application. Peaks show some deadlines for deliverables in each platform, starting with the PC and SB platforms, followed by HMDJ and 3DD. We can also notice that the later versions required less changes (or less time) to complete than the previous ones. This indicates that the cost of producing new versions is reduced during time.

The last peaks in the *All/Common* curve are related to the log mechanism of the applications. All versions log information about the user's experience in a centralized database, and this functionality required several changes, during time.

Table 4.1 shows the average of changes per platform, during the total period of time considered in Figure 4.4 (190 days). Although one can not compare such numbers due to the diminishing effort given by the succession of development cycles, we can see that the HMDJ and 3DD versions have a higher change average than the other two platforms. Analyzing such a result we have found that this effect is due to the several versions of InTml files we created for testing such environments, and due to the special classes we created to deal with the special devices that these platforms required. We can also see that the SMARTBoard environment was really easy to integrate, despite the special hardware it involved. Again, we can notice that most of the effort was concentrated on common files to all platforms.

Platform	Average in LOC changes
PC	9.6
SB	7.3
HMDJ	20.5
3DD	16.2
All	98.2

Table 4.1: Average number of Lines of Code (LOC) changes per Platform.

4.4 Comparison with Other Development Methodologies and 3D Technologies

Other methodologies for the development of 3D applications have been proposed in the last decade. The methodology described in this section differs from previous proposals in the level of abstraction that designers handle, and the special management that retargeting requires. Our methodology works at the architectural level of the development cycle, and that the details in other methodologies can be used in order to achieve the expected results.

The work by Tanriverdi and Jacob [93] divides a 3D application in the following components: graphics, behavior, interaction, mediator, and communication. They define a two–level design process, by dividing high level and low level decisions in each component. Our methodology deals mostly with the high level decisions mentioned in this paper. Our methodology can be enriched with the low level design process for media elements (which corresponds to their graphics component), behavior (which is subdivided in their behavior, interaction, and mediator components), and application connectivity (which corresponds to their communication component).

Smith, Duke, and Massink [84] define a formal method to compare interaction techniques in different platforms, in a way independent from the actual code. While this notation is very useful to understand the inner behavior of an interaction technique, a development methodology based on such a notation can be too detailed for our designers. We believe that such a notation can be used for detailed design of complex interaction filters in our methodology, provided some mapping between their concept of continuous devices and our strictly discrete modeling.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Stanney et al. [88] define a set of usability guides for the design of VR applications. We believe that such guidelines can be easily used and integrated to our proposal, in the task of describing applications in InTml documents. At this stage designers should take into consideration not only user requirements but also usability guidelines that can be applied to the targeted hardware platforms.

There are also some differences between the InTml–based development versus development based on other technologies, such as VRML or VRJuggler. We think that our methodology and division of tasks between roles can be applied with other implementation environments, but there are some limitations.

First, an architectural language is required for designers, in order to allow them to develop the same tasks at a higher level of abstraction than code. There are no other architectural languages specially targeted to VR applications, and general–purpose architectural languages can not show the important issues in a 3D application.

Second, environment capabilities can be an issue in a general interactive application. For example, VRML is an environment for 3D applications where users can navigate environments, but their interactive capabilities are limited: i.e. it is not possible to allow an object to be translated in the three axes and rotated at the same time; several independent modes have to be created. Such limitations are very important when an implementation environment is chosen *a priori*, without any consideration to the application requirements.

Third, since current VR environments do not provide a clear description of behavioral components and their interrelationships [15], developers can decide any subdivision as they wish, precluding our support for retargeting. For example, code for reading devices is usually intermingled with the expected response to it, so it is difficult to replace such devices without going to the level of code.

Fourth, retargeting is a new development tasks defined in our methodology, and depends heavily in our division of components. Without this clear separation is not possible to retarget an application by changing components, and more detail changes at the code level are required.

As one can see, our methodology separates from particular implementation mechanisms in order to create more portable, retargetable applications. It is also heavily supported by our language, InTml.

[15]Java3D does have clearly defined behavioral components, but interrelationships are not so clear since all behaviors have to be awaken by the system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

Examples of Use

In this chapter we demonstrate with three examples how the concepts presented in this thesis can be used. The first is a simple application that we developed as a proof of concept for our retargeting process. The second is a comparative user interface study between several hardware platforms, based on the simple application of our example in the previous chapter. This user study shows how different implementations in different hardware platforms can be compared, and how our development techniques help in the comparison and development processes of multi–platform VR applications. The final example is a partial design, that was developed with Pauline Jepp, a PhD student at the University of Calgary, that shows how designers can start the development of a complex application for several hardware platforms by using InTml.

5.1 A Matching Application

We have already mentioned this application in Section 4.2, as an example of the development methodology for InTml applications. We will describe in detail the components behind the application's tasks, and their interrelationships. The InTml files of the four implementations of this application are in Appendix A.

The application runs on different hardware platforms:

- The PC aplication uses a standard PC interface (Figure 5.1), with a CRT monitor, a mouse and a keyboard.
- The SMART Board based (SB) application uses a SMART Board [94] (Figure 5.2). Our SMART Board is a front projected system with touch–sensible screen, four pens, and an eraser. There are also two buttons beside the pen holders, but we did not use them in this application.
- The HMDJ application uses a Virtual I-O Glasses head mounted display [50] and a Microsoft Sidewinder force–feedback joystick [31] (Figure 5.3). The I-O Glasses is a low resolution HMD with a 3 degrees–of–freedom (DOF) tracker for head orientation. The SideWinder has many capabilities, but we only use its positioning device and four buttons.
- The Space Mouse based PC (3DD) application uses a CRT monitor, a keyboard, and a SpaceMouse input device [1] (Figure 5.4). The SpaceMouse is a 6 DOF device that allows users to change the position and orientation of a given object. It also has some buttons integrated in the device, but we use the keyboard instead for implementation reasons.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

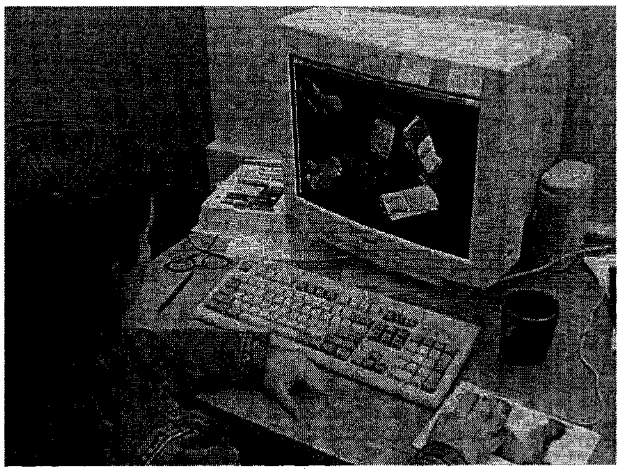


Figure 5.1: Application running on a PC environment.

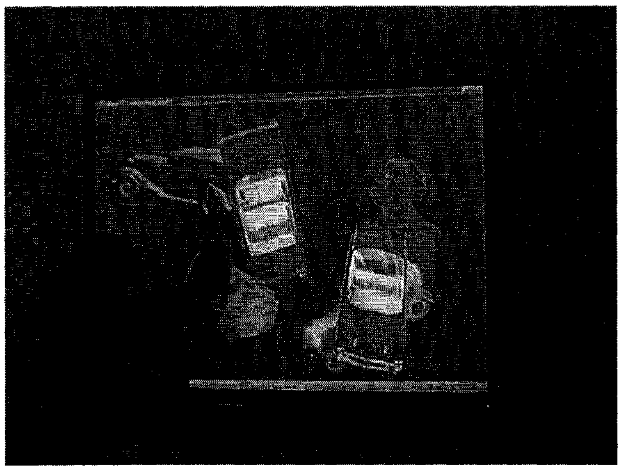


Figure 5.2: Application running on a SMART Board.

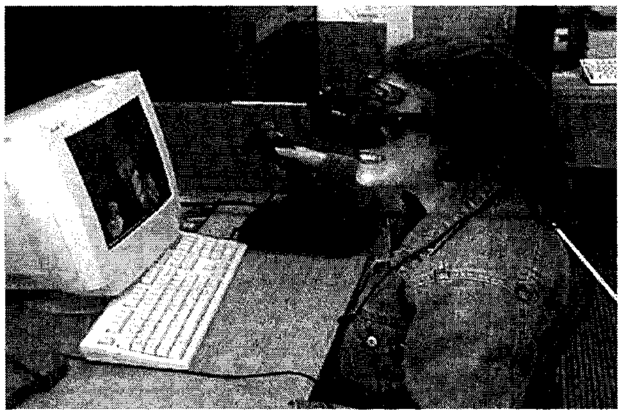


Figure 5.3: Application running on a HMD plus Joystick.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

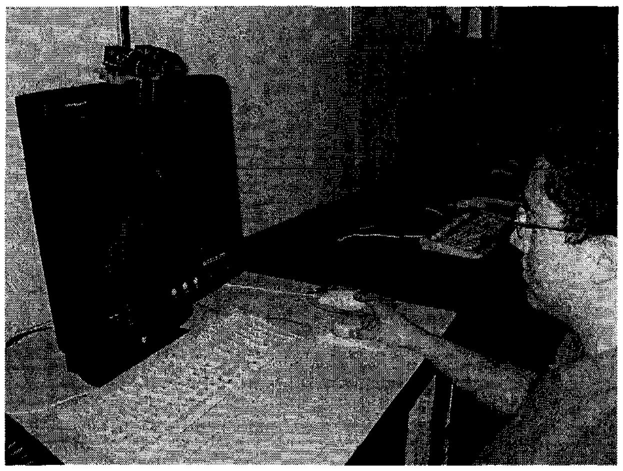


Figure 5.4: Application running on a PC plus 3D Mouse.

The Matching application was designed as a self–contained set of InTml files, without references to external libraries, since InTml general libraries of devices and interaction techniques were also under development. For this reason, the application defines all devices and interaction techniques required by its implementation.

The following sections describe the InTml implementation of each task in the application. Such tasks were previously mentioned in Section 4.2.

Object's Geometry Loader

Object's geometry is loaded from external files, in OBJ and 3ds formats. The InTml code for loading the three objects in the application is in the Listing 16[1].

Listing 16 Load objects in an InTml Application.

```
<Object id="obj1" filename="media/car.3ds" type="VRObject"/>
<Object id="obj2" filename="media/Dodge32.3ds" type="VRObject"/>
<Object id="obj3" filename="media/beethoven.obj" type="VRObject"/>
```

Each object is named by an unique identifier. Geometry is loaded from a file, given here by a path relative to the application location. The object's type defines what functionality is expected. In this case, we use the `VRObject`, which was described in Section 3.3. The code in Java loads each file into memory, without making them visible. The implementation of this task is platform independent.

Localizing and Orienting Objects at Random

The filter RandomPQ assigns a random position and orientation to the objects it receives. Its interface is shown in Figure 5.5.

[1]The XML notation in this section's listings was described in Section 3.4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

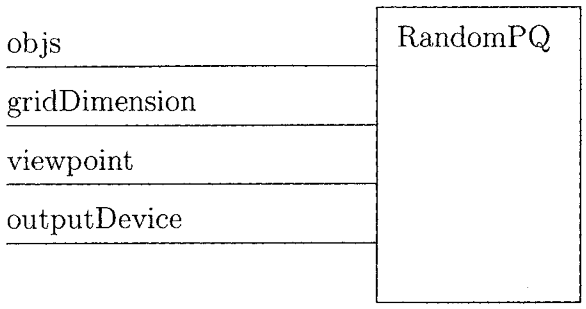


Figure 5.5: RandomPQ ports.

The filter divides the space in front of the viewpoint in a grid of size `gridDimension`. The `outputDevice` parameter is used to change the distance at which the objects are shown, according to the type of display (HMD or a normal monitor). Consecutive objects received through the port `objs` are set to a random position in the grid. Extra parameters are available from the Java implementation of the class, to define a near and far clipping planes for the random position generation, and a random seed. In general, such parameters are interesting to developers only, not to designers, and we will not discuss this further.

During initial testing of the application, we noticed that some objects were closer to their copies than others, and that some objects occlude others since they were closer in our perspective projection. This is because `RandomPQ` does not take into consideration the relative distances (euclidean and angular) between an object and its copy, and the way the screen is divided. We decided to create a more specialized class, `RandomRelativePQ`. This new class gets an object and its copy at the same time, divides the screen in 6 disjoint areas, assigns a random position to the object in one of these areas, and assigns a random position to the object's copy such that the angular and euclidean distance between an object and its copy are constant. The new class is shown in Figure 5.6. Each object and its copy are received in different ports, so they can be distinguished. Two parameters are necessary, the output device to use, and the viewpoint position and orientation.

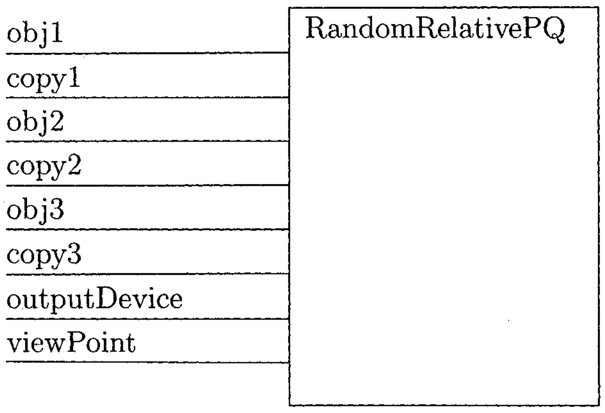


Figure 5.6: RandomRelativePQ ports.

Listing 17 shows the InTml code that creates the filter of class `RandomRelativePQ`, and connects the objects and parameters to it. The previous code for `RandomPQ` was similar, with the difference that all objects were received in just one port. This code can be considered platform independent, providing that the output device is called `screen` in all

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

implementations.

Listing 17 Localize objects at random.

```
<Filter id="randomRelativePQ" type="RandomRelativePQ"/>

<Binding iE="_self" iP="obj1" oE="randomRelativePQ" oP="obj1"/>
<Binding iE="_self" iP="obj2" oE="randomRelativePQ" oP="obj2"/>
<Binding iE="_self" iP="obj3" oE="randomRelativePQ" oP="obj3"/>
  <!-- Bind the current output display -->
<Binding iE="_self" iP="screen" oE="randomRelativePQ" oP="outputDevice"/>

  <!-- Bind the current viewpoint -->
<Binding iE="_self" iP="viewpoint" oE="randomRelativePQ" oP="viewPoint"/>
```

Creating Copies of Objects and Localizing Them at Random

Copies of objects are created by the filter `TransparentCopy`. It has one input port that receives objects and one output port that produces transparent copies of the received objects. Listing 18 shows the InTml code that creates the copies, and send them to the interested filters. We create three instances of `TransparentCopy` since we want to distinguish each object's copy. Copies of objects are sent to `randomRelativePQ` in order to change their initial position and orientation to the one required by the application. This task is platform independent.

Listing 18 Create copies of objects and localize them.

```
<Filter id="tCopy1" type="TransparentCopy"/>
<Filter id="tCopy2" type="TransparentCopy"/>
<Filter id="tCopy3" type="TransparentCopy"/>
<Binding iE="_self" iP="obj1" oE="tCopy1" oP="obj"/>
<Binding iE="_self" iP="obj2" oE="tCopy2" oP="obj"/>
<Binding iE="_self" iP="obj3" oE="tCopy3" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="randomRelativePQ" oP="copy1"/>
<Binding iE="tCopy2" iP="objCopied" oE="randomRelativePQ" oP="copy2"/>
<Binding iE="tCopy3" iP="objCopied" oE="randomRelativePQ" oP="copy3"/>
```

Defining Selectable Objects

We designed filters for the different selection techniques that we used. The PC, HMDJ, and SB versions use `SelectByRay` in different ways, while 3DD uses `SelectByTouching`. The simplest selection technique is `SelectByRay`. This technique selects an object from a 2D position in the screen, which defines a ray perpendicular to the user's plane of view. In its simplest form, it has two input ports for the 2D position and the scene of selectable objects, and an output port for the current selected object (Figure 5.7). It also has extra ports for control purposes, inherited from `ControlableFilter`, but they are not shown in the figure since they are not relevant for the actual selection technique.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

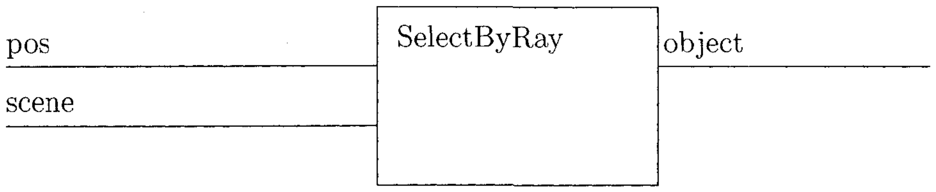


Figure 5.7: SelectByRay ports.

The PC version of the selection task is straightforward (Listing 19): Objects are added to the set `selectableObjs`, the selection technique is created and its input is plugged into the set of selectable objects and the 2D position from the mouse [2].

Listing 19 Define selectable objects, PC version.

```
<Object id="selectableObjs" type="Scene"/>
<Binding iE="_self" iP="obj1" oE="selectableObjs" oP="addObject"/>
<Binding iE="_self" iP="obj2" oE="selectableObjs" oP="addObject"/>
<Binding iE="_self" iP="obj3" oE="selectableObjs" oP="addObject"/>

<Filter id="selection" type="SelectByRay"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
<Binding iE="mouse" iP="mousePos" oE="selection" oP="pos"/>
```

The SB version uses the same interaction technique, but instead of receiving a 2D position from the mouse, it receives information from the touch–sensitive whiteboard. Listing 20 shows the new code [3], with the connection of the SMARTBoard output to the selection technique. There is a difference with the PC example that is worth mentioning: The SMARTBoard allows users to detach from the input device, to jump from one position in the screen to another, which is different from the mouse that always computes a position as a relative displacement from the previous position. For this reason, we added functionality to allow the interaction technique to know when users are not touching the screen, so the selected object can be de–selected. Without this functionality, users select an object, leave the screen, and see the object still in selected mode. It is also important to remember that filters react to any user input in their input ports, but not to absence of input. A selection technique can not react to absence of a position for selection, since without an input it does not have to be executed at all.

Listing 20 Define selectable objects, SB version.

```
<Filter id="selection" type="SelectByRay"/>
<Binding iE="smartboard" iP="touchPos" oE="selection" oP="pos"/>
<Binding iE="smartboard" iP="screenReleased"
        oE="selection" oP="flushState"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
```

The HMDJ version takes into account two more input devices available: A simple orientation tracker attached to the HMD and a joystick. The interaction technique created takes

[2]The class `Scene` allows us to create sets of objects that are not objects themselves, so they can not be selectable.

[3]Since the creation of the set of selectable objects is the same, the code is not repeated here and in the following listings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

into consideration both devices as it is shown in Figure 5.8. The user's head movements define the current viewpoint and view plane as a displacement of the real one. Since we do not use the mouse pointer for interaction, a new visual representation is created. Also, since we do not want to lose the pointer once the user's head moves, the pointer position is defined in terms of the current view plane center and the actual joystick displacement. In this way, the pointer is always visible, and it can be moved by the joystick movements or indirectly by head movements [4].

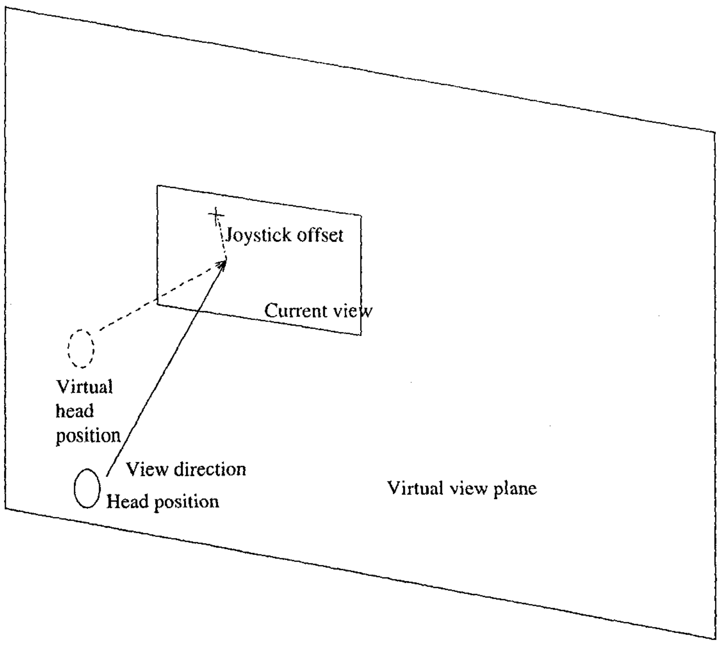


Figure 5.8: View and Selection in the HMDJ Platform.

The HMDJ version of the selection technique, illustrated in Listing 21, has a filter to compute the center of the view plane (`center`, of type `OrientationCenter`) [5], a filter to compute the joystick pointer position (`pointerPos`, of type `HMDJPointer`), and a visual representation of the pointer (`pointer`). The `center` filter requires the head's position (a constant in this case), the head's orientation, and the current output device in order to compute the view plane center. The most important input ports for `pointerPos` are the current view plane center and the joystick position. The final step in this setup is to change the actual position of the pointer representation with the 3D position computed by `pointerPos`. The selection technique takes the 2D position computed by `pointerPos` and the scene. We can see that we do not actually change the code for the selection technique, but the actual setup.

Finally, the selection in the 3DD environment is based on collision with a hand representation, as shown in Listing 22. The filter class `SelectByTouching` (Figure 5.9) performs this task, which requires the selectable objects, the hand representation object, and a signal when to compute such a collision (everytime the hand representation is moved or rotated). The hand position and orientation comes from the `mouse3D` device [6].

[4]The joystick might appear fixed relative to the user's view plane, but it is moving in absolute coordinates.
[5]We can notice that the view is an approximation of the real view. It can be improved in order to compute the frustrum from the real head position.
[6]The position actually comes from the `resetPosition` filter, that allows users to move the cursor back to the center of the screen, but this functionality is out of this scope of this description.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 21 Define selectable objects, HMDJ version.

```
<!-- Show selection pointer from the head tracker plus joystick -->
<Filter id="center" type="OrientationCenter"/>
<Filter id="pointerPos" type="HMDJPointer"/>
<Constant id="shift" type="float" value="0.01"/>
<Object id="pointer" filename="media/pointer.obj" type="VRObject"/>
<Binding iE="_self" iP="pV" oE="center" oP="p"/>
<Binding iE="tracker" iP="q" oE="center" oP="q"/>
<Binding iE="_self" iP="hmd" oE="center" oP="screen"/>
<Binding iE="center" iP="pos" oE="pointerPos" oP="headCenter"/>
<Binding iE="joystick" iP="pos" oE="pointerPos" oP="jPos"/>
<Binding iE="_self" iP="hmd" oE="pointerPos" oP="screen"/>
<Binding iE="_self" iP="shift" oE="pointerPos" oP="shiftAmount"/>
<Binding iE="pointerPos" iP="pos3D" oE="pointer" oP="setPos"/>

<!-- Create selection technique and bind it as necessary -->
<Filter id="selection" type="SelectByRay"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
<Binding iE="pointerPos" iP="pos" oE="selection" oP="pos"/>
```

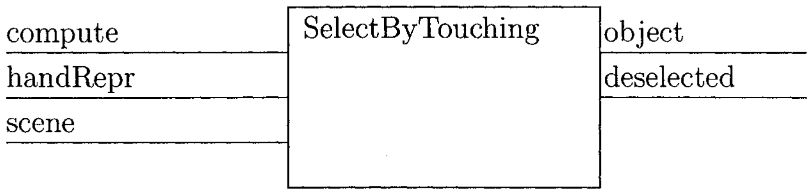


Figure 5.9: SelectByTouching ports.

Listing 22 Define selectable objects, 3DD version.

```
<Filter id="selection" type="SelectByTouching"/>
<Object id="vHand" filename="media/pointer.obj" type="VRObject"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
<Binding iE="mouse3D" iP="pos" oE="selection" oP="compute"/>
<Binding iE="mouse3D" iP="q" oE="selection" oP="compute"/>
<Binding iE="_self" iP="vHand" oE="selection" oP="handRepr"/>
<Binding iE="resetPosition" iP="pos" oE="vHand" oP="setPos"/>
<Binding iE="mouse3D" iP="q" oE="vHand" oP="setQ"/>
```

Feedback for The User Selection

Objects are shown in green when they are selected, in all platforms. The InTml code for this functionality is shown in Listing 23. The filter `highlight` changes the color of an object to the selected color and back to the normal one, once an event by the `deselected` port is received or a new object is selected.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 23 Define selectable objects, 3DD version.

```
<Filter id="highlight" type="HighlightedFeedback"/>
<Binding iE="selection" iP="object" oE="highlight" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="highlight" oP="deselected"/>
```

Grabing and Releasing an Object

Objects are grabbed in order to start a manipulation by translations or rotations. Each platform uses different devices and interaction techniques for this functionality, as follows.

The PC version implements object rotation and translation with mouse dragging operations, left and right buttons. In this case, the grabbing operation is different either for rotation or translation. The SB version uses also separate dragging controls with four different pens (one for rotation, three for translation). The HMDJ based implementation uses four buttons in the joystick for grabbing an object, either for rotation or translation. The 3DD platform takes a different approach since it is possible to rotate and translate an object at the same time, with the same device. In addition, we decided to offer a toggle functionality for both rotation and translation. In this way, it is possible to just move or just rotate by pressing a key. Users found this very useful in order to divide the task in a succession of rotations and translations.

The code related to this functionality is shown in the following section, including the code for rotations and translations.

Rotating and Translating an Object

An object can be rotated or translated in several ways, given a certain set of input devices. We decided to implement simple interaction techniques that could be reused among platforms. In the future, these interaction techniques can be replaced by more standard ones, especially for rotations [47].

The PC, SB, and HMDJ versions use the `RotationBehavior` and `TranslationBehavior` classes for rotating and moving objects, shown in Figure 5.10. `RotationBehavior` takes the unit of rotation per pointer movement, starting events, ending events, the current pointer position, the object to be rotated, and an indication that the object is not selected anymore. `TranslationBehavior` takes a movement scale, starting events, ending events, the current pointer position, the object, and an indication of when an object is deselected. It also takes input events that define in which plane the movement will be applied (XY/XZ/YZ), from the user's viewpoint.

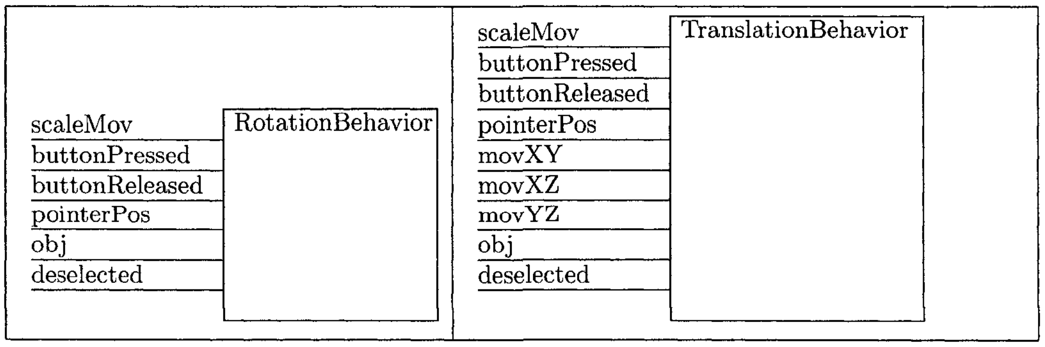


Figure 5.10: RotationBehavior and TranslationBehavior ports.

The PC version of these manipulation techniques shown in Listing 24 [7], connects the

[7]In the following Listings, we replace `translateObj` for `tObj` and `joystick` for `j` for space reasons

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

rotation technique to left–button dragging in the mouse, and translation to right–button dragging. In order to switch between the three axes of movement, we use the keys z, x, and c. We thought of using the standard letters for coordinate axes, but we decided to choose three consecutive letters instead, since users can just leave their fingers on top of them and select them, without looking at the keyboard.

Listing 24 Rotate and translate objects, PC version.

```
<!-- Rotate an object -->
<Filter id="rotateObj" type="RotationBehavior"/>
<Binding iE="mouse" iP="lButtonPressed"
         oE="rotateObj" oP="buttonPressed"/>
<Binding iE="mouse" iP="lButtonReleased"
         oE="rotateObj" oP="buttonReleased"/>
<Binding iE="mouse" iP="mousePos" oE="rotateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="rotateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotateObj" oP="deselected"/>

<!-- Translate an object -->
<Filter id="tObj" type="TranslationBehavior"/>
<Binding iE="mouse" iP="rButtonPressed"
         oE="tObj" oP="buttonPressed"/>
<Binding iE="mouse" iP="rButtonReleased"
         oE="tObj" oP="buttonReleased"/>
<Binding iE="mouse" iP="mousePos" oE="tObj" oP="pointerPos"/>
<Binding iE="keyboard" iP="z" oE="tObj" oP="movXY"/>
<Binding iE="keyboard" iP="x" oE="tObj" oP="movXZ"/>
<Binding iE="keyboard" iP="c" oE="tObj" oP="movYZ"/>
<Binding iE="selection" iP="object" oE="tObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="tObj" oP="deselected"/>
```

The SB version connects rotation to the first pen in the SMARTBoard, and each translation axis to pens 2, 3, and 4, for planes XY, XZ, and YZ, respectively. The code is shown in Listing 25. We can see how the selection of a pen for translation both activates the translation technique and selects the particular plane of movement.

Listing 26 shows the code for rotation and translation in the HMDJ platform. This code is similar to the one of in the SB platform, except that four buttons in the joystick are used for movements and rotations.

Our last platform, 3DD uses a different class for rotation and translation, due to the fact that the space mouse allows users to perform both operations at the same time. Listing 27 shows the corresponding code, in which keys z and x are used for grabbing and releasing the selected object, and keys a and s are used to switch on and off translation and rotation, respectively. Another difference between this implementation and the previous ones is that it is possible to get a 3D rotation and translation directly from the device. RotTrans takes this into account and maps more naturally events from the device into object movements.

Computing Matching Function

The matching between an object and its copy is a platform–independent operation that measures their angular and euclidean distances and generate an event when both distances are under certain thresholds. The class MatchFunction takes an object, its copy, and an event of when it has to be computed. It generates an event when both objects are close enough.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 25 Rotate and translate objects, SB version.

```
<!-- Rotate an object -->
<Filter id="rotateObj" type="RotationBehavior"/>
<Binding iE="smartboard" iP="pen1Selected"
         oE="rotateObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen1Released"
         oE="rotateObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="touchPos" oE="rotateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="rotateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotateObj" oP="deselected"/>

<!-- Translate an object -->
<Filter id="tObj" type="TranslationBehavior"/>
<Binding iE="smartboard" iP="pen2Selected"
         oE="tObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen2Released"
         oE="tObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="pen2Selected" oE="tObj" oP="movXY"/>
<Binding iE="smartboard" iP="pen3Selected" oE="tObj" oP="movXZ"/>
<Binding iE="smartboard" iP="pen3Selected"
         oE="tObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen3Released"
         oE="tObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="pen4Selected" oE="tObj" oP="movYZ"/>
<Binding iE="smartboard" iP="pen4Selected"
         oE="tObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen4Released"
         oE="tObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="touchPos" oE="tObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="tObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="tObj" oP="deselected"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 26 Rotate and translate objects, HMDJ version.

```
<!-- Rotate an object -->
<Filter id="rotateObj" type="RotationBehavior"/>
<Binding iE="j" iP="button1Pressed" oE="rotateObj" oP="buttonPressed"/>
<Binding iE="j" iP="button1Released" oE="rotateObj" oP="buttonReleased"/>
<Binding iE="j" iP="button1Pressed" oE="pointerPos" oP="buttonPressed"/>
<Binding iE="j" iP="button1Released" oE="pointerPos" oP="buttonReleased"/>
<Binding iE="pointerPos" iP="pos" oE="rotateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="rotateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotateObj" oP="deselected"/>

<!-- Translate an object -->
<Filter id="tObj" type="TranslationBehavior"/>
<Binding iE="j" iP="button2Pressed" oE="tObj" oP="buttonPressed"/>
<Binding iE="j" iP="button2Released" oE="tObj" oP="buttonReleased"/>
<Binding iE="j" iP="button2Pressed" oE="pointerPos" oP="buttonPressed"/>
<Binding iE="j" iP="button2Released" oE="pointerPos" oP="buttonReleased"/>
<Binding iE="j" iP="button3Pressed" oE="tObj" oP="buttonPressed"/>
<Binding iE="j" iP="button3Released" oE="tObj" oP="buttonReleased"/>
<Binding iE="j" iP="button3Pressed" oE="pointerPos" oP="buttonPressed"/>
<Binding iE="j" iP="button3Released" oE="pointerPos" oP="buttonReleased"/>
<Binding iE="j" iP="button4Pressed" oE="tObj" oP="buttonPressed"/>
<Binding iE="j" iP="button4Released" oE="tObj" oP="buttonReleased"/>
<Binding iE="j" iP="button4Pressed" oE="pointerPos" oP="buttonPressed"/>
<Binding iE="j" iP="button4Released" oE="pointerPos" oP="buttonReleased"/>
<Binding iE="pointerPos" iP="pos" oE="tObj" oP="pointerPos"/>
<Binding iE="j" iP="button2Pressed" oE="tObj" oP="movXY"/>
<Binding iE="j" iP="button3Pressed" oE="tObj" oP="movXZ"/>
<Binding iE="j" iP="button4Pressed" oE="tObj" oP="movYZ"/>
<Binding iE="selection" iP="object" oE="tObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="tObj" oP="deselected"/>
```

Listing 27 Rotate and translate objects, 3DD version.

```
<!-- Rotate and Translate an object -->
<Filter id="rotTrans" type="RotTrans"/>
<Binding iE="selection" iP="object" oE="rotTrans" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotTrans" oP="deselected"/>
<Binding iE="keyboard" iP="z" oE="rotTrans" oP="buttonPressed"/>
<Binding iE="keyboard" iP="x" oE="rotTrans" oP="buttonReleased"/>
<Binding iE="keyboard" iP="a" oE="rotTrans" oP="toggleTranslation"/>
<Binding iE="keyboard" iP="s" oE="rotTrans" oP="toggleRotation"/>
<Binding iE="mouse3D" iP="q" oE="rotTrans" oP="setQ"/>
<Binding iE="mouse3D" iP="pos" oE="rotTrans" oP="setPos"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 28 Matching function.

```
<Filter id="matchFunction1" type="MatchFunction"/>
<Filter id="matchFunction2" type="MatchFunction"/>
<Filter id="matchFunction3" type="MatchFunction"/>
<Binding iE="_self" iP="obj1" oE="matchFunction1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="matchFunction1" oP="copyObj"/>
<Binding iE="_self" iP="obj2" oE="matchFunction2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="matchFunction2" oP="copyObj"/>
<Binding iE="_self" iP="obj3" oE="matchFunction3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="matchFunction3" oP="copyObj"/>
<Binding iE="obj1" iP="posChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj1" iP="qChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj2" iP="posChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj2" iP="qChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj3" iP="posChanged" oE="matchFunction3" oP="compute"/>
<Binding iE="obj3" iP="qChanged" oE="matchFunction3" oP="compute"/>
```

Deleting Objects Once They are Matched

We make objects invisible in order to simulate the operation of deleting. The filter `DeleteWhenSignal` sends a signal to the object and its copy received through its input port `obj`, once an event through its `signal` port is received. As illustrated in Listing 29, we create three instances of this filter in order to delete each tuple object–copy.

Listing 29 Deleting objects function.

```
<Filter id="deleteObjs1" type="DeleteWhenSignal"/>
<Filter id="deleteObjs2" type="DeleteWhenSignal"/>
<Filter id="deleteObjs3" type="DeleteWhenSignal"/>
<Binding iE="_self" iP="obj1" oE="deleteObjs1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="deleteObjs1" oP="obj"/>
<Binding iE="matchFunction1" iP="match" oE="deleteObjs1" oP="signal"/>
<Binding iE="_self" iP="obj2" oE="deleteObjs2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="deleteObjs2" oP="obj"/>
<Binding iE="matchFunction2" iP="match" oE="deleteObjs2" oP="signal"/>
<Binding iE="_self" iP="obj3" oE="deleteObjs3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="deleteObjs3" oP="obj"/>
<Binding iE="matchFunction3" iP="match" oE="deleteObjs3" oP="signal"/>
```

Logging Application Events

We logged all events that result of the interaction with the application, in all four implementations. Figure 5.11 shows the input ports for the Log filter class, which basically registers all possible output events: time associated with each execution frame, user and platform identifications, matching signals, end or abort signals, changes in positions and orientations for objects and object copies, selection signals, change of plane of movemement (XY/XZ/YZ), starting and ending signals for rotation and translation, and toggle signals for rotation and translation. Not all events can be generated in just one system; instead, each implementation connects ports to an instance of the Log class according to the information available.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

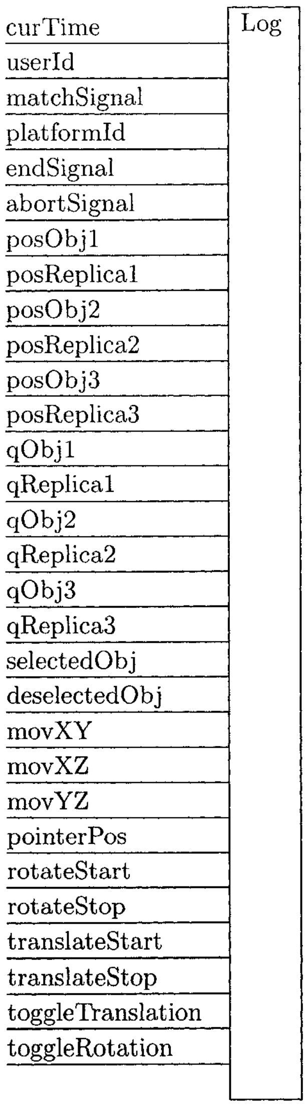


Figure 5.11: Log ports.

Listing 30 shows the connections for logging on the PC platform. The frame time is taken from the timer device. The user id and platform id are initialized with constant values. With the aid of a program written in XSLT [108], the user id was replaced by a different value each time the program was run. Events from the mouse, keyboard, matching filters, selection filter, and objects are sent to the log accordingly. Copies of objects are wrapped in object holders, and their position and orientation changes are routed to the log.

The SB version of the log functionality is similar to the PC version, with the differences that the devices generate from information. Listing 31 shows such differences. At a difference of the PC version, we can notice that there are several ways to start a translation, since there are different devices for each translation plane.

The version for the HMDJ version is similar to the SB one, and it can be found in Appendix A. Listing 32 shows the differences in the 3DD version. In this case, keys are associated to the task of grabbing an object for rotation and translation. There are also keys for toggling rotation and translation behavior, which is not present in the previous versions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 30 Logging functionality on the PC platform.

```
<Filter id="log" type="Log"/>
<IDevice id="timer" type="Timer"/>
<Binding iE="timer" iP="curTime" oE="log" oP="curTime"/>
<Binding iE="quit" iP="endInfo" oE="log" oP="endSignal"/>

<!-- Identify user and platform -->
<!-- Change this constant to an id for a particular user -->
<Constant id="userId" type="String" value="Test user"/>
<Constant id="platformId" type="String" value="PC"/>
<Binding iE="_self" iP="userId" oE="log" oP="userId"/>
<Binding iE="_self" iP="platformId" oE="log" oP="platformId"/>

<!-- initial transformations, selected objects, position and orientation
     while moving, match times -->
<Binding iE="matchFunction1" iP="match" oE="log" oP="matchSignal"/>
<Binding iE="matchFunction2" iP="match" oE="log" oP="matchSignal"/>
<Binding iE="matchFunction3" iP="match" oE="log" oP="matchSignal"/>
<Binding iE="mouse" iP="mousePos" oE="log" oP="pointerPos"/>
<Binding iE="mouse" iP="lButtonPressed" oE="log" oP="rotateStart"/>
<Binding iE="mouse" iP="lButtonReleased" oE="log" oP="rotateStop"/>
<Binding iE="mouse" iP="rButtonPressed" oE="log" oP="translateStart"/>
<Binding iE="mouse" iP="rButtonReleased" oE="log" oP="translateStop"/>
<Binding iE="keyboard" iP="z" oE="log" oP="movXY"/>
<Binding iE="keyboard" iP="x" oE="log" oP="movXZ"/>
<Binding iE="keyboard" iP="c" oE="log" oP="movYZ"/>
<Binding iE="keyboard" iP="q" oE="log" oP="abortSignal"/>
<Binding iE="selection" iP="object" oE="log" oP="selectedObj"/>
<Binding iE="selection" iP="deselected" oE="log" oP="deselectedObj"/>

<Binding iE="obj1" iP="posChanged" oE="log" oP="posObj1"/>
<Binding iE="obj1" iP="qChanged" oE="log" oP="qObj1"/>
<Binding iE="obj2" iP="posChanged" oE="log" oP="posObj2"/>
<Binding iE="obj2" iP="qChanged" oE="log" oP="qObj2"/>
<Binding iE="obj3" iP="posChanged" oE="log" oP="posObj3"/>
<Binding iE="obj3" iP="qChanged" oE="log" oP="qObj3"/>

<ObjectHolder id="copy1"/>
<ObjectHolder id="copy2"/>
<ObjectHolder id="copy3"/>
<Binding iE="tCopy1" iP="objCopied" oE="copy1" oP="object"/>
<Binding iE="tCopy2" iP="objCopied" oE="copy2" oP="object"/>
<Binding iE="tCopy3" iP="objCopied" oE="copy3" oP="object"/>
<Binding iE="copy1" iP="posChanged" oE="log" oP="posReplica1"/>
<Binding iE="copy1" iP="qChanged" oE="log" oP="qReplica1"/>
<Binding iE="copy2" iP="posChanged" oE="log" oP="posReplica2"/>
<Binding iE="copy2" iP="qChanged" oE="log" oP="qReplica2"/>
<Binding iE="copy3" iP="posChanged" oE="log" oP="posReplica3"/>
<Binding iE="copy3" iP="qChanged" oE="log" oP="qReplica3"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 31 Logging functionality on the SB platform.

```
<Binding iE="smartboard" iP="touchPos" oE="log" oP="pointerPos"/>
<Binding iE="smartboard" iP="pen1Selected" oE="log" oP="rotateStart"/>
<Binding iE="smartboard" iP="pen1Released" oE="log" oP="rotateStop"/>
<Binding iE="smartboard" iP="pen1Selected" oE="log" oP="translateStart"/>
<Binding iE="smartboard" iP="pen1Released" oE="log" oP="translateStop"/>
<Binding iE="smartboard" iP="pen2Selected" oE="log" oP="translateStart"/>
<Binding iE="smartboard" iP="pen2Released" oE="log" oP="translateStop"/>
<Binding iE="smartboard" iP="pen3Selected" oE="log" oP="translateStart"/>
<Binding iE="smartboard" iP="pen3Released" oE="log" oP="translateStop"/>
<Binding iE="smartboard" iP="pen4Selected" oE="log" oP="translateStart"/>
<Binding iE="smartboard" iP="pen4Released" oE="log" oP="translateStop"/>
<Binding iE="smartboard" iP="pen2Selected" oE="log" oP="movXY"/>
<Binding iE="smartboard" iP="pen3Selected" oE="log" oP="movXZ"/>
<Binding iE="smartboard" iP="pen4Selected" oE="log" oP="movYZ"/>
```

Listing 32 Logging functionality on the 3DD platform.

```
<Binding iE="keyboard" iP="z" oE="log" oP="rotateStart"/>
<Binding iE="keyboard" iP="x" oE="log" oP="rotateStop"/>
<Binding iE="keyboard" iP="z" oE="log" oP="translateStart"/>
<Binding iE="keyboard" iP="x" oE="log" oP="translateStop"/>
<Binding iE="keyboard" iP="q" oE="log" oP="abortSignal"/>
<Binding iE="keyboard" iP="a" oE="log" oP="toggleTranslation"/>
<Binding iE="keyboard" iP="s" oE="log" oP="toggleRotation"/>
```

Terminating the Application

There are two ways to end the application, either by completing the task of matching the three objects, or by aborting. Both are implemented in a platform–independent way as showed in Listing 33. The filter class `QuitMatching` ends the application when it receives either three matching signals or one abort signal.

Listing 33 InTml code for ending the application.

```
<Filter id="quit" type="QuitMatching"/>
<Binding iE="matchFunction1" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction2" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction3" iP="match" oE="quit" oP="signal"/>
<Binding iE="keyboard" iP="q" oE="quit" oP="abortSignal"/>
```

5.2 Performance Comparison for The Four VR Platforms

The following example shows how InTml can help in the design and development of comparative user studies. In any user study, developers have to create different versions of a test application, in order to cover the different conditions they want to explore. InTml facilitated the creation of such versions, since it is trivial to change filters as part of the retargeting process. Although this study is not very conclusive, it illustrates a powerful aspect of InTml, which is the facilities to create variations of a same application in comparative user studies. We conducted an experiment in which subjects used one of four versions

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of the matching application running on different hardware platforms. We used InTml as the implementation framework in order to share as many similarities as possible between different hardware platforms and, at the same time, to be able to retarget the interaction techniques to the available hardware. InTml also facilitated the creation of common metrics among all implementations.

5.2.1 Comparison Metrics

Many user studies in VR, mentioned later, have concentrated on the systematic analysis of interaction techniques, devices, and content characteristics. Some have studied variations of a particular application, and some have concentrated on the reactions that people have to this technology. Our goal is to define metrics and methods to compare all these factors at once, for a particular application and a user community. The following sections present a detailed analysis of these studies.

Several studies have compared interaction techniques, such as comparisons of manipulation techniques [72, 109], grabbing techniques [16], and travel techniques [19]. Such comparisons usually create environments with parameters relevant to a particular set of interaction techniques in study, and they typically use a fixed set of devices. Other studies have compared a smaller subset of interaction techniques in order to show advantages and disadvantages in more detail, such as a virtual hand versus a virtual pointer [70], *HOMER* versus *Vodoo Dolls* [69], or speed-coupled flying with orbiting versus others [91].

Several papers have introduced new VR devices (e.g. [80, 46]), some with detailed user studies about their benefits and characteristics (e.g [65, 60, 101]), and some with comparison with other devices in a particular task [47]. Other papers have been devoted to the study of small changes in devices for an application, such as a joystick versus a stylus [17], a 3D–ball versus a tracker [47], two hand control versus one hand [7], or eye movements versus a pointer [92]. Although these studies provide good design guidelines for VR developments, they are of limited use in real applications when a device is not used in isolation.

Some papers have shown studies of several implementations of interesting VR applications in areas such as 3D model design [43] or non–linear drama [32], but they have considered only a limited number of implementations, and their analysis techniques have been very subjective.

There are also some studies on the effect of VR environments on humans, e.g. a study of object rotation [102], a study of role of kinesthetic reference frames in two handed input performance [6], or a study on nausea effects of navigation techniques [48]. These studies found design guidelines that are very useful in the development of VR applications.

We argue that the important issue in a real application is the combination of all factors, interaction techniques, devices, and people. Hence, it is difficult to directly apply previous studies since they do not show the effect of factor combinations. Previous results are very important as guidelines, but a set of previously tested devices and interaction techniques have to be tested together in order to understand their relationships.

We use objective and subjective metrics to compare several versions of an application. There are many ways to capture subjective information from the user, but we decided to use questionnaires, because they can be filled online and they are easily processed. There are many examples of questionnaires for user evaluation in VR applications, e.g. [106, 83], and the more general ones are applicable to any interface [22]. Despite known disadvantages, e.g. a lack of correlation with reality [99], we consider them useful for comparing different implementations of a VR application.

We collected the following standard set of events for all platforms:

- SEL and DES: Selection and de-selection of objects.
- MOV_XY, MOV_XZ, MOV_YZ: Change in the plane of movement, applicable in all platforms except 3DD.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- RI and RE: Subject starts or stops an object rotation.
- TI and TE: Subject starts or stops an object movement.
- INTML_EXEC_TIME: Time for the execution of tasks in InTml, in milliseconds.
- MATCH_SIGNAL: Event received when an object and its copy are close enough so they are considered a match.
- END_SIGNAL and ABORT_SIGNAL: Events received when the application finishes or its aborted.
- MOV: An object has moved
- ROT: An object has been rotated

Some events can not be generalized, such as the MOV_* group, since such events do not exist in the 3DD platform. Others behave differently in different platforms, such as the RI/RE/TI/TE: In the 3DD platform, both groups are simultaneous to the SEL–DES events, since once an object is grabbed, it can be rotated and translated at the same time [8]. Still these events allow the development of a set of uniform metrics and an objective comparison of different implementations. We define the following metrics, based on the raw events described previously: Time for object matching, number of control events per object, distance error, orientation error, and preparation time. Details of such metrics are described below.

The events generated by an application are temporally organized as shown in Figure 5.12. Objects are manipulated in selection intervals, the intervals that start with a SEL and end with a DES. We define the time for object matching as the sum of all selection intervals dedicated to a particular object, intervals that might be intermixed and sparse, but are always disjoint. The number of control events per object is the sum of all control events, MOV_XY, MOV_XZ, MOV_YZ, RI, RE, TI, TE, inside the selection intervals of an object.

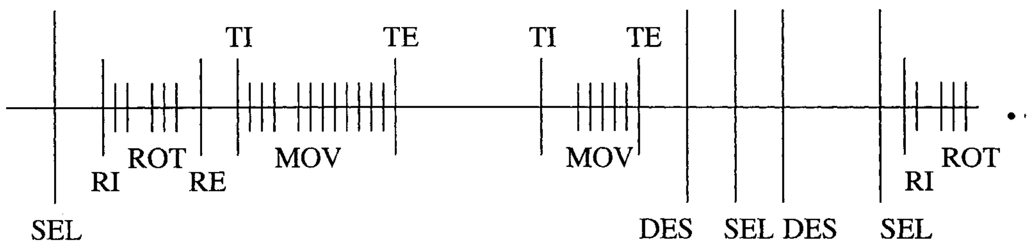


Figure 5.12: Order of events throughout time.

The distance error (DE) is defined in terms of the MOV events. Every time such an event is generated for a particular object, we save the distance to its copy, and we consider this distance fixed during the period of time given by the MOV event and the next one, or the end of the selection interval. The distance error can vary substantially, as is illustrated in Figure 5.13. Ideally, the user's interaction will steadily decrease the distance between an object and its copy. However, errors in the user's interpretation or in the use of the interface can create a more erratic behavior, as in the right panel of the figure. We take the number of times that the distance increases as a measure of the distance error in an experiment. This measure can be computed in several ways, depending on how fine–grain detail changes are considered, and how important the errors are. In this analysis, we use every third point, and we record any increase in distance. All other measure techniques we used gave similar

[8]We also use these groups to mark the tasks' dis-entanglement in the 3DD environment.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

results. The orientation error (OE) is defined in a similar way, except that every time a ROT event is received, we consider the angle between the quaternions that describe the current orientation of an object and its copy.

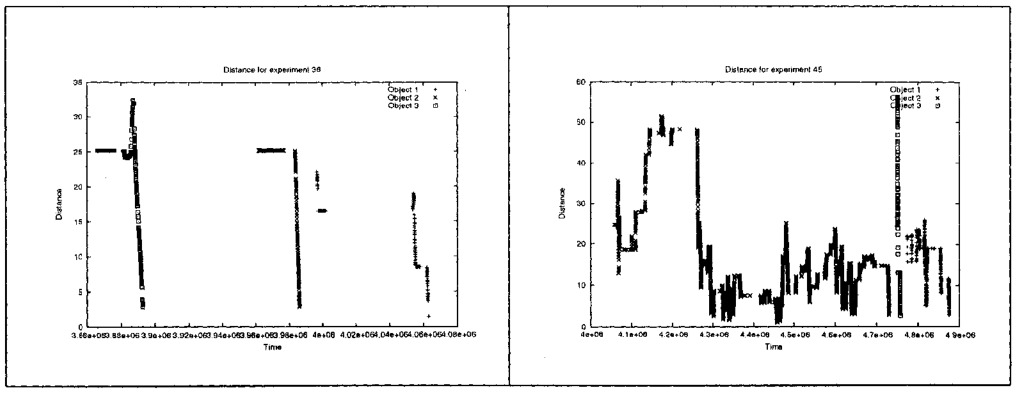


Figure 5.13: Distance error over time.

If there are no objects selected, we assume that the user is preparing an interaction with the system. The preparation time related to an object is the sum of all preparation intervals that precede selection intervals for this object.

5.2.2 Method and Apparatus

We conducted an experiment in which subjects used one of four versions of an application running on different hardware platforms. A total of 42 participants volunteered for the experiment, 8 women and 34 men between 21 and 30 years. They were divided into 4 groups, one for each hardware setup. Three participants did not finish the experiment, and we did not consider their data in the analysis, so there were 10 participants for all experimental condition except one. Participants read a general introduction to the experiment and filled out a form with general information about themselves, inspired by the QUIS evaluation questionnaire [82, 22] and shown in Appendix E. We guided them while they were matching the first object, and allowed them to be free with the other two objects.

After the experiment was completed, performance data was collected and the participants were asked to fill out a second form, a short version of the one designed by Witmer and Singer [106] (see Appendix E) about their experiences and personal opinions. Participants took about half an hour to complete the whole experiment.

5.2.3 Results

We will discuss in this section the general results of the experiment, the event–related results, and the subjective results of the questionnaire evaluation.

General Results

Table 5.1 shows the InTml frame rate, the time spent in an execution of InTml behavior, given by the average of execution time of the InTml code, and the Java3D frame rates, given by the mean of the difference between start times of InTml frames. We find that the InTml execution rates for the SB and HMDJ platforms are considerably slower than for the PC and 3DD. The InTml frame rates for the PC and 3DD environments are enough to keep the Java3D frame rates up to speed. The implementation of the SB and HMDJ environments should, however, be faster in order to get at least one update per Java3D frame.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Platform	InTml Frm. R.	Java3D Frm. R.
PC	51	50.65
SB	29	57.33
HMDJ	31	34.83
3DD	60	28.68

Table 5.1: InTml and Java3D Frame Rates.

Table 5.2 shows the average duration and standard errors [9] per experiment in each platform. PC took longer than the other platforms, and HMDJ was the fastest one, despite the lower frame rate that this platform had.

Platform	Time (min)	SE
PC	11.9	2.46
SB	10.76	1.03
HMDJ	7.61	0.91
3DD	9.75	0.93

Table 5.2: Time per Experiment.

Event–Related Results

Most users started the task with the Beethoven face, matched second the yellow car and the red car as the last one.

Table 5.3 shows the time for object matching, in seconds, for the 2nd and 3rd matched objects for each platform. We notice that matching times do indeed decrease between the second and third matching. From faster to slower matching time, we have the following order of platforms: 3DD, SB, HMDJ, PC. It is interesting to notice the good performance of the SB platform, despite the slower hardware that it uses. The PC platform performs slower than the others, despite the fact that users were already familiar with the devices used in this platform.

Platform	2nd Object		3rd Object	
	MEAN	SE	MEAN	SE
PC	262.6	103.0	134.8	29.4
SB	119.3	18.9	100.8	23.8
HMDJ	129.2	36.3	128.2	25.4
3DD	106.8	27.2	98.1	22.1

Table 5.3: Time for matching.

Table 5.4 shows the number of control events per platform. 3DD presents the lowest number of control events, mainly due to the absence of MOV_* events. Despite the uniformity of interaction techniques in the SB and PC platforms, SB registers significantly fewer control events than PC.

[9] The standard error is defined as the standard deviation divided by the square root of the number of samples.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Platform	2nd Object		3rd Object	
	MEAN	SE	MEAN	SE
PC	940.9	549.8	619.3	186.6
SB	63.3	11.8	62.6	10.5
HMDJ	26.5	3.8	26.9	2.9
3DD	4.2	2.5	3.8	1.8

Table 5.4: Number of control events.

Table 5.5 shows distance errors. Users are more accurate in the SB and HMDJ platforms than in the PC and 3DD ones. Pairwise differences between SB and 3DD, and between HMDJ and 3DD are significant.

Platform	2nd Object		3rd Object	
	MEAN	SE	MEAN	SE
PC	20.7	11.3	10.1	2.6
SB	3.5	1.0	2.7	0.9
HMDJ	5.3	0.9	5.9	1.4
3DD	13.6	3.2	19.3	2.4

Table 5.5: Distance error.

Table 5.6 shows orientation errors. Users are more accurate in the SB and the 3DD platforms than in the PC and HMDJ ones. The difference between the PC and 3DD platforms for the second object is significant.

Platform	2nd Object		3rd Object	
	MEAN	SE	MEAN	SE
PC	76.4	25.9	52.5	15.1
SB	15.7	3.4	15.9	3.1
HMDJ	28.2	14.6	36.9	6.1
3DD	13.1	4.3	23.1	4.7

Table 5.6: Orientation error.

Finally, Table 5.7 shows the preparation time per platform. The HMDJ platform requires more time before selecting objects, followed by 3DD, SB, and PC.

Platform	2nd Object		3rd Object	
	MEAN	SE	MEAN	SE
PC	16.4	3.7	11.8	2.1
SB	104.9	13.9	92.4	17.8
HMDJ	22.0	5.2	29.6	6.0
3DD	94.7	35.4	68.5	25.9

Table 5.7: Preparation time.

Table 5.8 shows rankings for all platforms, according to each one of the above measures. There are some ties due to contradictory results between the second and third matched object. On average, the HMDJ, SB, and 3DD platforms outperform the PC. Again, it is interesting to note the good performance of the SB platform despite the slower hardware.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Platform	TOM	NCE	DE	OE	TT	AVE
PC	4	4	3.5	4	4	3.9
SB	2	3	1	1.5	3	2.1
HMDJ	3	2	2	1.5	1	1.9
3DD	1	1	3.5	3	2	2.1

Table 5.8: Rankings of platforms per metric.

Subjective Results

Users are classified according to two factors: immersive tendency and previous 3D experience. We consider that participants have had previous experience with 3D systems if they selected any of the following options in the first questionnaire (Appendix E): Game consoles, HMDs, Tracking devices, Stereo glasses, Passive Stereo Displays, 3D Mouse, CAD, or 3D Video Games. Table 5.9 shows the averages of immersive tendency and previous experience.

Platform	Immersive Tend.	3D experience
PC	6.05 ± 0.47	90%
SB	5.5 ± 0.37	80%
HMDJ	5.97 ± 0.38	90%
3DD	5.7 ± 0.51	89%

Table 5.9: Participants' previous experiences.

Table 5.10 shows average answers of questionnaries.

Platform	User Reactions	Screen	Learning	System Capabilities	Sense of Presence
PC	5.4 ± 0.31	8.33 ± 0.08	5.12 ± 0.13	7.1 ± 0.44	5.73 ± 0.4
SB	5.58 ± 0.17	7.73 ± 0.16	5.84 ± 0.29	5.23 ± 0.42	5.59 ± 0.35
HMDJ	6.08 ± 0.19	6.97 ± 0.28	5.52 ± 0.25	6.45 ± 0.23	6.41 ± 0.34
3DD	6.41 ± 0.33	7.26 ± 0.19	5.6 ± 0.1	7.08 ± 0.18	5.76 ± 0.53

Table 5.10: Averages of user answers in the surveys.

User reactions to the PC and SB environments are not as good as the ones for HMDJ and 3DD. Some users complained about the difficulty of the interaction techniques in the PC and SB environments, especially for rotations. Users in the HMDJ platform asked to map rotations in the Z axis to the joystick's twist capability. Users liked the screen appearance of the PC best, and that of the HMDJ least. Users complained about the screen size in the HMDJ environment, and about the visual feedback for selectable objects, especially in the 3DD environment [10]. In terms of learning, users did not find any interface particularly intuitive, and the most disliked was the PC platform. The system as a whole was considered better for participants in 3DD and PC environments than for participants in the SB and HMDJ ones. Finally, presence ratings were in general very low, with a slight advantage for the HMDJ platform. Users complained about the interaction techniques, the front projection system in the SB, the mapping of the joystick in the HMDJ, and the jittering of the tracker in the HMDJ.

A similar analysis of Table 5.8, with the information from the users, reveals that all platforms are similar from the user's point of view, with a slight preference for the SB environment. It is also interesting to note that despite the fact that the PC and HMDJ

[10]In the 3DD environment, users check if an object is selectable by moving the pointer to the object's position, which is slow.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

environments are comparable from the user's point of view, the HMDJ has a better objective ranking. For this reason, both objective and subjective metrics should be taken into account in the development of new prototypes.

5.2.4 Discussion

These four prototypes show some interesting results. The frame rates were low in SB, as can be expected with the slower machine use. HMDJ also had a low InTml and Java3D frame rates, possibly caused by device drivers, orientation tracker or display. Users in the HMDJ platform completed the task faster despite its low frame rate, which makes this platform very promising in terms of user performance. Matching times for the 3DD and SB were comparable and better than the ones in PC and HMDJ. Users took probably more time to prepare in these environments, and they were more efficient when they were interacting. This result is also supported by the number of control events generated in the PC platform compared to the others: PC users generated many more control events than users of the other platforms. We believe that users were more comfortable and familiar with the PC platform, so they allowed themselves more interaction mistakes than in the other environments.

Users produced large distance errors in the PC and 3DD platforms than in the other two. We believe that they took more time to interact and move the objects in these two environments, and consequently the distance errors were bigger. There is a bigger number of orientation errors than distance errors, suggesting that users spent most of the time rotating objects, instead of moving them.

Users spent more time preparing in the SB environment than in the other ones, suggesting that the characteristics of this environment, bigger display and slower machine, affected their performance. However, increased preparation time resulted in lower number of errors and shorter matching times.

From the questionnaires, we see that users reacted more critically to the PC environment than to the others. Maybe the familiarity of this environment created a higher expectations with respect to the interaction techniques. Screen quality was better on the PC, but it is interesting that users gave a lower rating to the 3DD, despite the fact that they had basically the same screen. The lack of familiarity with the Space Mouse may have affected the overall rating of image quality.

The PC and 3DD platorms were classified as more powerful than the HMDJ and SB. We believe this is due latency and delays on the latter two and the better machines used for the former two. Users in 3DD complained about lack of visual feedback, despite the fact that all platforms used the same approach: the copy of an object was not selectable, only the original, so any attempt to select a copy did not give any feedback. We believe this is due to the slower process of selecting an object in the 3DD platform, since users have to move a pointer in 3D until it collides with an object. This can take more time than simply moving a 2D pointer in the display plane and using ray casting for selection, so the slower the process the more noticeable were problems with feedback. Finally, users rated the HMDJ better in terms of presence, maybe because vision with the HMD was limited to the virtual world, so distractions were avoided.

We have shown how several implementations of a 3D application can be developed and compared using objective and subjective data. Our results show that the environment that users prefer may not necessarily be the one in which they perform best. The results also show that implementations on slow machines can compete with faster ones, at least at a prototype level for the interaction techniques and behavior. In summary, the development of several prototypes of an application, instrumented by standard metrics among them, can help understanding the best implementation for a given population.

We plan to run more studies with different applications, in order to test our methodology more thoroughly. We also plan to add more platforms to this study, to do more studies about controlled improvements of this application, and to compare specially tuned implementations

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in these platforms.

5.3 A Virtual Clay Application

The collaboration with Pauline Jepp at the University of Calgary evolved in a first InTml description of tasks in her Virtual Clay Application. The Virtual Clay Application is a multi–platform, VR environment that will allow artists to create models with virtual clay, a virtual material with similar characteristics to real clay. InTml code for the tasks we have designed is included in Appendix B, and we describe here the current state of our design effort.

We envisioned three platforms where the application should run:

- PC: A standard desktop computer, with a monitor, mouse, and keyboard.
- Simple3D: A platform with active stereo display [11], and a SpaceBall [25] as input device.
- Haptic3D: A platform with active stereo display and a Phantom [76] device for input and force–feedback.

The functionality that the application will provide includes the following tasks:

- Load models. A way to save and load models will be provided. What is particularly important for this task is that the model should be deformable as clay, and this affects the type of information to be saved.
- Navigate around the model. The application will allow users to see the model in construction from different viewpoints. The most common navigation will consist on rotating the model around its principal axis, which is the way artists analyze clay models in the real world. More navigation techniques can be provided, according to device affordances.
- Manipulate modeling tools. The way artists manipulate clay in the real world is by deforming a piece of material with a set of objects, or tools, which imprint certain shape and texture to the piece. The application will allow users to grab tools and use them to re–shape the virtual clay.
- Spin material. A common way to model clay is by mounting a piece in a wheel and spinning it, so an artist can define its contour. The virtual clay application should allow artists to spin a virtual material, given a certain axis of rotation. Tools can be used in this mode in order to shape the contour of the clay piece.
- Contact feedback. If force–feedback is available, users should receive some feedback from the manipulation they perform on the clay.
- Cut material. A clay piece can be deformed with tools, and it can also be cut. The virtual clay application will allow both modes of operation. If force–feedback is available, a cut will be performed if enough force is applied in the interaction with the virtual material.
- Undo the application. This operation will provide some degree of undo, in order to allow users to return to previous states of their work.

[11]We call active stereo to the stereo technique of showing consecutive images to each eye, which are covered by shutter glasses.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We are in the initial design stages of this application, So far, we have identified classes for specialized devices in the Simple3D and Haptic3D platforms, and some classes for the implementation of the navigation task in the PC platform. The special devices are a Space Ball [25] input device, and a force–feedback Phantom arm [76]. The required interface for the space ball is similar to the one of a Space Mouse, with a difference in the number of buttons available. The required interface for a Phantom device is shown in Figure 5.14. Forces and torques can be defined at any given time. The device keeps a list of objects to collide with, that can be modified. The output consists in the current position and orientation of the tip of the arm. This is a partial definition of the Phantom capabilities, and we plan to extend them once we define in more detail the application for the Haptic3D platform.

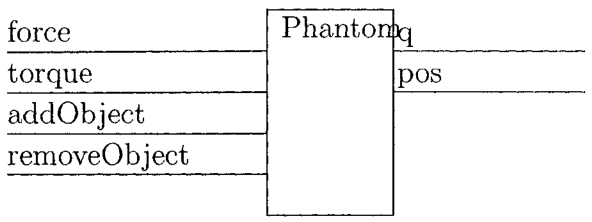


Figure 5.14: Phantom ports.

The classes related to the PC implementation are [12]:

- `Material`: Material models, i.e. any material that can be deformed by or deform to any other object.
- `VirtualClayEnvironment`: The environment that coordinates deformation among objects.
- `Mouse2Ray`: The mouse position defines a ray in the current screen.
- `IntersectByBSphere`: Intersect with objects by using theirs bounding sphere.
- `RotateByDragging`: Rotates an object by using a point in its bounding sphere.
- `TranslatePlane`: Moves an object in a plane, given a 2D position. The input position may come from a pointer, as a mouse or a joystick

The application uses the RotateByDragging technique in order to rotate the clay material in the PC environment. The position of the mouse in the screen is translated into a 3D ray that can be collided with an object. Dragging with the left mouse button rotates a point in the bounding sphere that contains the object, once a collision has been detected. Releasing the left mouse button releases the object, and flushes the state of the interaction technique. The InTml code for such a behavior is shown in Listing 34.

While our current state of development does not have retargeted tasks, we are confident that InTml will allow us to describe this application in the identified platforms, and to reuse as much code as possible.

[12]These filter classes are defined in the file `newClasses.intml`, available in Appendix B.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 34 InTml code for rotation in the PC–based virtual clay application.

```
<Filter id="mouse2Ray" type="Mouse2Ray"/>
<Filter id="getPoint" type="IntersectByBSphere"/>
<Filter id="rotate" type="RotateByDragging"/>
<Filter id="switchRot" type="Switch"/>
<Binding iE="mouse" iP="mousePos" oE="mouse2Ray" oP="pos2D"/>
<Binding iE="theCurrentViewpoint" iP="objectChanged"
              oE="mouse2Ray" oP="viewpoint"/>
<Binding iE="_self" iP="screen" oE="mouse2Ray" oP="screen"/>
<Binding iE="_self" iP="clay" oE="getPoint" oP="object"/>
<Binding iE="mouse2Ray" iP="ray" oE="getPoint" oP="ray"/>
<Binding iE="clay" iP="currentBSphere" oE="getPoint" oP="compute"/>
<Binding iE="getPoint" iP="point" oE="rotate" oP="point"/>
<Binding iE="_self" iP="clay" oE="rotate" oP="object"/>
<Binding iE="mouse" iP="lButtonPressed" oE="switch" oP="signalOn"/>
<Binding iE="mouse" iP="lButtonReleased" oE="switch" oP="signalOff"/>
<Binding iE="switch" iP="onOff" oE="rotate" oP="on"/>
<Binding iE="switch" iP="onOff" oE="rotate" oP="flushState"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 6

Support Tools for InTml–Based Development

In this section, we describe the general architecture of our InTml–based development environment, and the tools developed within this architecture. The tools are in a first prototype stage, more coverage of the InTml support and more work on the user interface are required. Nevertheless, we consider that the current state shows the opportunities and development power of our two–level of abstraction technique, and the possibilities of a more user–friendly environment. We have concentrated our work on the following tools: a library of interaction techniques, a browser for InTml documents, an InTml checker, and InTml compilers into Java3D and C++. The following is a discussion of each one of these tools, preceded by a general overview of the InTml–based development environment.

6.1 An InTml–Based Development Environment

InTml describes VR applications and 3D interaction techniques that can be implemented in several toolkits and environments, such as Java3D, VRJuggler, or VRML/X3D. InTml supports two levels of abstraction during development, one suitable for designers with no programming skills, and one for developers with a good understanding of the implementation of components required by designers. Figure 6.1 shows our architecture for a development environment based on InTml. Division of tasks is encouraged by its architecture: VR designers define new applications by plugging-in components from the library, and VR developers write the required code for new components. The required code might be geometry and objects appearance details, code in the core framework for new interaction techniques and animations, or code for the integration of new devices. VR designers create new applications by composing components from the InTml library. We have envisioned the following tools for designers: a library browser, a semantic checker for new component definitions, a compiler that translates applications written in InTml to a particular framework language, and a visual programming environment. VR developers use, extend, and maintain a white box framework[1], that implements the InTml concepts on top of a core framework or API, and the implemented components of the InTml library.

[1]A white box framework is the first stage in the evolution of a framework as is defined in [74].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

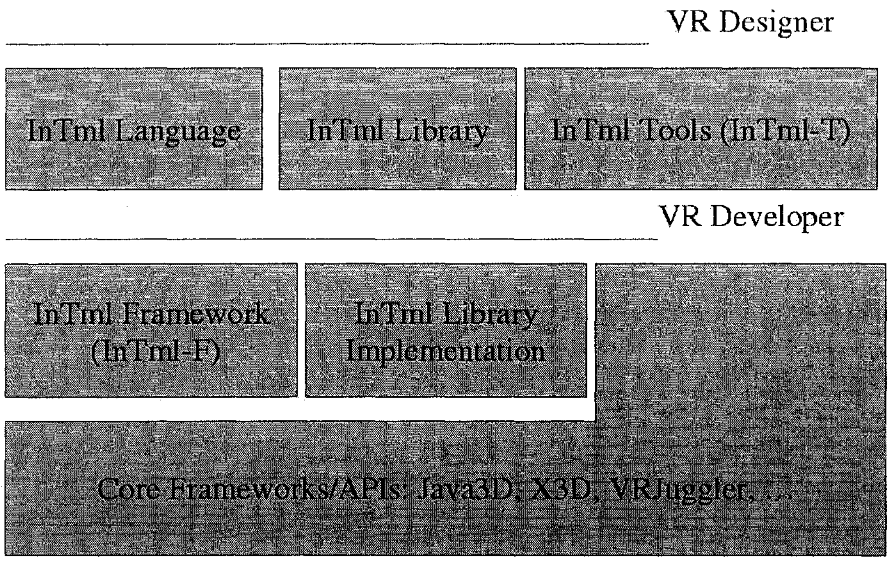

VR Designer
InTml Language
InTml Library
InTml Tools (InTml-T)
VR Developer
InTml Framework (InTml-F)
InTml Library Implementation
Core Frameworks/APIs: Java3D, X3D, VRJuggler, ...

Figure 6.1: Architecture of an implementation of InTml.

A new implementation of InTml, in terms of Figure 6.1, consists of creating the appropriate compiler, InTml Framework, and InTml Library Implementation for a particular Core Framework. For example, an implementation of InTml in X3D can be defined as a set of rules of how prototypes for interaction techniques should be defined, the InTml library implementation can be a set of prototypes that implement the interfaces defined in the InTml library, and the compiler can be embedded in a browser that reads both InTml and X3D files, or a stand-alone module that outputs X3D files from InTml descriptions. However, restrictions on the X3D execution model may compromise performance in such an implementation. For this reason, despite that InTml can be implemented on many platforms, some of them may perform better and support more features than others.

The following sections describe in more detail current implementations of this development environment architecture, and the future developments we have envisioned.

6.2 The InTml Language

We have already described the current state of the InTml language in Section 3.4. We describe here the DTD that defines it.

The first version of InTml is described in a Data Type Definition File for XML, or DTD[2]. It defines all elements required for InTml applications and for creating libraries of devices, interaction techniques, or objects. Definitions can be divided in three main groups: InTml entities, mechanisms of reuse, and documentation. The entire DTD is found in Appendix C.

[2]We could use also XML Schema for this purpose, but at the time we started this implementation the XML Schema specification was not entirely approved.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Some DTD-specific mechanisms were used in order to make such a description more concise. For example, sets of attributes are enclosed in ENTITY clauses, which are macro substitutions into elements that use them. In this way, several elements reuse the definition of an identifier (ID), an identifier with required type (ID_TYPEREQ), or attributes for arrays of ports (ARRAY_DEF). DTD files have a special attribute type ID, which allows the definition of unique identifiers per file. We decided not to use this feature of a standard DTD due to the fact that we need identifiers of several types in the same and across files, and the DTD mechanism for IDs assumes both a file as a boundary and that there is only one type of IDs.

The InTml elements described in the DTD can be divided into two main groups: class elements and instance elements. The following paragraphs describe these two subcategories[3].

Class elements help designers to create new classes of entities in the InTml environment: new devices, new platforms, new filters, or new objects. The DTD elements DeviceClass, PlatformClass, FilterClass, and ObjectClass define such types. A DeviceClass defines new types of devices in the environment, in terms of the input and output ports available. A PlatformClass defines a collection of input and output devices, that can be used together in a particular application. A FilterClass defines either a simple filter, with just input and output ports, or a composed one with internal filters, objects, object holders, constants, and connections between them. Finally, an ObjectClass defines the input and output ports expected from a particular type of object in the environment.

Instance elements describe instances of InTml classes inside composed filters or applications [4]. Filter defines an instance of a FilterClass, given a name and a type. Object defines an instance of an ObjectClass, given a name, type, and some geometry [5]. An ObjectHolder defines a new object reference, whose input and output ports are defined by the first object to be held. Binding describes a connection between InTml instances (filters, devices, objects) in terms of a name of an input element, its output port, and a corresponding name of an output element with a compatible input port. IPort and OPort define input and output ports inside InTml instances. They allow the definition of arrays of ports, with either fixed or dynamic size [6]. App defines all elements in an application: A platform, objects, constants, object holders, filters, and bindings. ODevice and IDevice define output and input devices, from a certain DeviceClass. Constant defines a constant in an InTml application. In this way, initial values are sent to ports in an application. Finally, a Platform is an instance of a certain PlatformClass

Four mechanisms of reuse have been designed in the InTml language. Package defines a name space for a set of classes and applications. It avoids name conflicts between classes and gives structure to a library of InTml concepts. Import includes definitions from other files to the current one. Implements is an interface–based inheritance mechanism: all input and output ports defined in an implemented filter are added to the current definition. This mechanism is similar to the mechanism with the same name in Java. Overrides is a form of delegation between filter classes. If class B overrides A, B contains all ports defined in A that are not defined in B, plus the ones in B. This mechanism allows us to create applications as enhanced versions of previous ones, by overriding some definitions and adding some more.

The InTml DTD defines some simple ways to document all previous elements. Almost all elements require a short description, defined by ShortDesc, and might have a longer one, with the element Description. InTml classes extracted from interaction techniques from other authors can add a paper reference, as a PaperRef, and their classification in one or several indexes, so library users can browse classes in a more semantic way.

[3]This information has been already presented in Section 3.4. It is presented here in a condensed form from the point of view of the DTD file.

[4]Currently, an application is defined just as an instance, not as a type

[5]Geometry can be loaded from an external file or defined as one of the following primitive types: Box, Cone, Cylinder, Ellipse

[6]A port with dynamic size allows other InTml instances to connect to a port they like, without any restriction of index. An example of when this is useful is a generic OR port, in which several boolean inputs are combined to produce a logical OR gate.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3 Library of Interaction Techniques

We use device classes to describe the input and output devices in our lab, and filter classes to describe some of the interaction techniques that have been proposed in the research community in the last 10 years. Full listings of the files in the library are included in Appendix D.

The available devices, defined in the file devices.xml are:

- `Button`: A basic button that sends a signal once is clicked.
- `GenericJoystick`: A basic joystick that outputs a 2D position
- `SideWinderPro`: Our InTml interface to a Microsoft SideWinder Pro joystick, implements the GenericJoystick interface and with 8 buttons (instances of the class Button), a hat switch as the 8 possible events it can generate (up, down, left, right, NE, SE, NW, SW), and two sliders for throttle and rotation.
- `Generic3DOFTracker`: A tracker that gives orientation only, as a quaternion.
- `Generic6DOFTracker`: Implements a Generic3DOFTracker, it adds a 3D position as output.
- `InsideTrack`: Implements a Generic6DOFTracker, our InsideTrak device.
- `InterSenseHeadTracker`: Implements a Generic6DOFTracker, our head tracker from InterSense.
- `InterSenseWandTracker`: Implements a Generic6DOFTracker and a Joystick. Includes also 4 buttons present in the InterSense Wand.
- `GenericScreen`: A generic 2D, PC screen.
- `FishTankScreen`: A 3D output screen, aware of the viewpoint position.
- `I-glasses`: Visual output of the I-glasses HMD.
- `VisroomScreen`: Each one of the screen in the Visroom, our CAVE–like environment.
- `GenericHMD`: A generic head mounted display description.
- `V6HMD`: Virtual Research's V6 HMD.
- GenericMouse: A general description of a mouse with three buttons and x and y positions.
- `WheelMouse`: A mouse with a wheel for scrolling.
- `GenericKeyboard`: A generic keyboard definition, with buttons that model letter keys.
- `MicroScribe3DFX`: Immersion's MicroScribe 3DFX 3d digitizer.

It is important to notice the separation that we do for abstract and concrete devices, i.e. `Generic6DOFTracker` and `InterSenseHeadTracker`. Both have the same interface, and in the current implementation the latter is defined completely in terms of the former, but they represent two different needs in InTml: the first description can be reused in other concrete devices and it can also represent common code in the implementation, while the latter allow designers to find a particular device easier.

Interaction techniques are divided in four groups: control, travel, selection, and the ones defined in the book by Barrileaux [8]. Control defines for the moment just one filter, QuitbyButton, that quits an application when a button signal is received as input. Travel defines the following filters:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- `SteeringIT`: Basic interaction technique associated with basic behavior in HMDs. It receives a position and orientation, and changes the viewpoint.
- `FlyUnrestrictedIT`: Moves the viewpoint as if one is flying around, given a starting point, a direction, and a certain speed.
- `FlyInPlaneIT`: Moves the viewpoint as if one is flying restricted to a plane.
- `WalkIT`: Moves the viewpoint as if one is walking.

Most of these interaction techniques are defined as composite filters, based on a simpler filter that defines the basic behavior, and a set of auxiliary filters and object holders that allow the changes to be made in the corresponding objects.

Selection techniques represented in InTml are:

- `FeedbackOneIT`: It changes the appearance of an object. It has memory, so the previously selected object restores its previous appearance once a new object is selected.
- `FeedbackSetIT`: The same as the previous one, but for a set of objects.
- `SelectByTouchingIT`: Selection by collision with an object that represents the user's pointer in the world (or her hand).
- `SelectByRayIT`: Selection in a particular direction.
- `GoGoIT`: Defined in [71], an interaction technique to lengthen the user's virtual arm for reaching distant objects
- `SpotlightIT`: It selects a set of objects, given a direction and a radius of a cone centered in such a direction [58].
- `RingMenuIT`: Selection technique defined in [58] that shows a ring of objects to the user, with the closest one as selected, and with an interior frame for readability purposes.

`SelectByTouchingIT`, `SelectByRayIT`, `GoGoIT`, and `RingMenuIT` are selection techniques that allow users to select just one object, whereas `SpotlightIT` allows multiple selection. This is why two feedback techniques are shown, one for just one object and one for groups. It is feasible to obtain multiple object selection techniques based on the basic behavior of the single–selection ones, but this is out of the scope of the current library.

The book by Barrileaux defines several interaction techniques. The following are a subset of those and some auxiliary definitions required.

- `DObject`: An object in display space. The display is the space that is in front of the user's view. It contains layers numbered from 1 (the closest).
- `WRM_MoveBasic`: Implements world-relative movement in a plane.
- `OrbitMovement`: Computes a new position and orientation around an object, given a starting position. It keeps the distance from the starting position to the center of the object.
- `OrbitCamera`: Implements navigation by moving around an object. It uses orbit movement in its implementation
- `PinocchioControl`: Computes the new position for the 3rd person control and for the viewpoint.
- `PinocchioCamera`: Implements navigation by manipulating a camera representation, with a body for position and a big nose for orientation in a plane.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- `ObjectBarrilleaux`: Describes an object with extended features, in order to support interaction techniques in Barrilleaux, such as identification of objects by name.
- `AddLabel`: Adds a label to an object. It assumes the object doesn't have one.
- `RemoveLabel`: Removes a label from an object. It assumes the object has one.

6.4 Tools

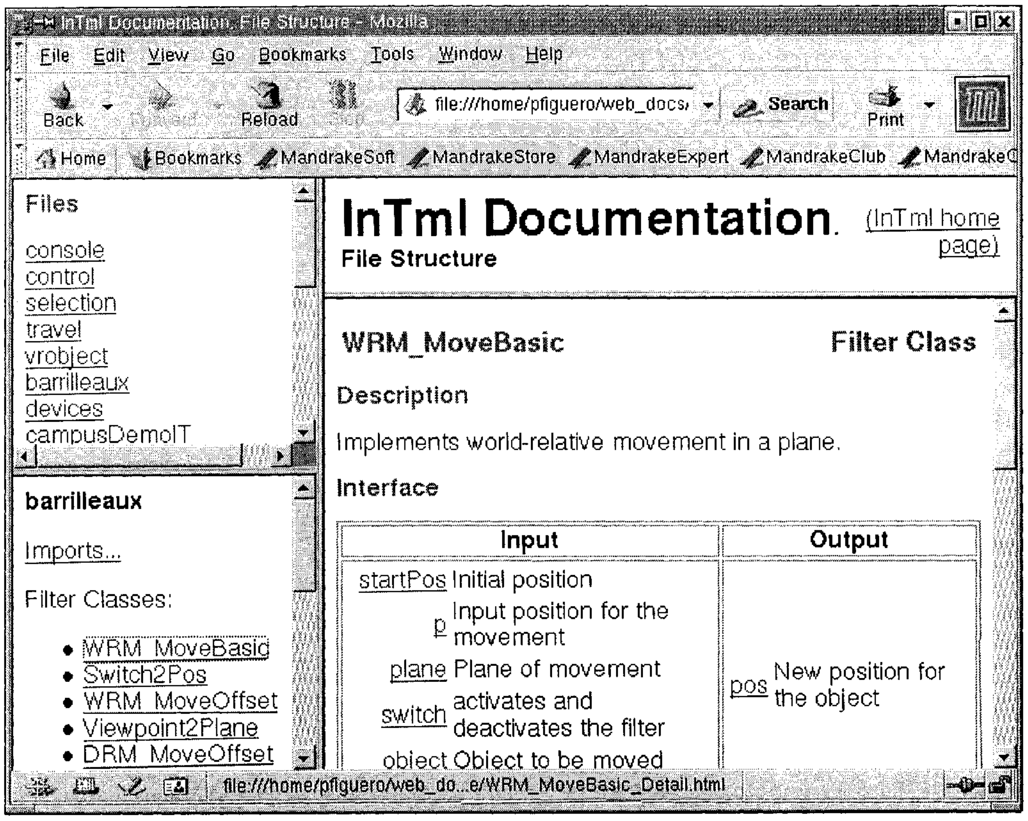
A professional development environment for InTml should count with a set of tools that allows designers and developers to be more efficient in their work. Modern programming languages usually count with a wide variety of integrated development environments (IDEs), in which developers find a set of tools that support coding tasks: editors, source code organizers, cross reference utilities, compilers, interpreters, documentation aids, and step–by–step debuggers. We envision something similar for InTml–based development, suitable for designers. Since developers create the inner details of components in core frameworks, they use an IDE suitable for such language. Some additional tools for designers are necessary, but we concentrate on the tools for designers, due to the fact that they are the ones less familiar with programming, and their work can be greatly benefited by the availability of such tools.

6.4.1 InTml Browser

The readability of InTml files, which are basically XML documents, can be greatly improved by the advantages that XML provides. XML documents can be shown in more readable formats by the definition of style files and translators in XSLT[108]. We have defined XSLT scripts that generate screens as the ones shown in Figures 6.2 and 6.3. This screen shows the following elements:

- File view. Applications and interaction techniques are classified by the file they are contained in. It is the basic view for the library contents and the simpler way to browse the information. Files are shown in the left-top frame of the screen. Once a file is selected, its contents are shown in the bottom left panel.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


InTml Documentation: File Structure - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Reload
file:///home/pfiguero/web_docs/
Search
Print
Home Bookmarks MandrakeSoft MandrakeStore MandrakeExpert MandrakeClub Mandrake
file:///home/pfiguero/web_do...e/WRM_MoveBasic_Detail.html

Files

console
control
selection
travel
vrobject
barrilleaux
devices
campusDemoIT

barrilleaux

Imports...

Filter Classes:

- WRM_MoveBasic
- Switch2Pos
- WRM_MoveOffset
- Viewpoint2Plane
- DRM_MoveOffset

InTml Documentation. (InTml home page)

File Structure

WRM_MoveBasic **Filter Class**

Description

Implements world-relative movement in a plane.

Interface

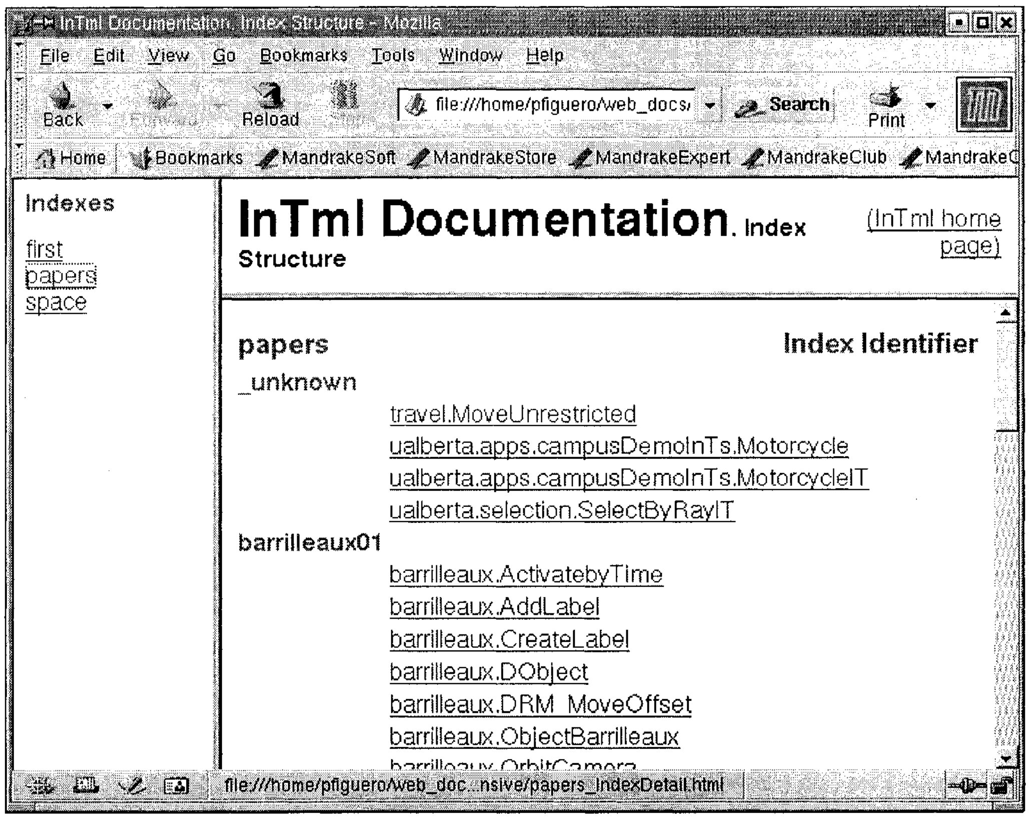
Input		Output	
startPos	Initial position	pos	New position for the object
p	Input position for the movement		
plane	Plane of movement		
switch	activates and deactivates the filter		
object	Object to be moved		

Figure 6.2: File view of InTml.

- Category views. It is possible to create several hierarchical indexes for filters and applications. In this way filters can be classified by several criteria, which can help designers to easily understand the library and choose the best options for a particular application. For example, all interaction techniques we have described in InTml are classified by the paper they appear in and by this basic classification: travel, selection, manipulation, control, and feedback [7]. A list of categories replace the list of files in the top–left corner of the screen. Once a category is selected, its categorization is shown in the main frame (right–bottom, in Figure 6.3).

[7]Inspired by Bowman's [18] and Barrileaux's [8] work

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


InTml Documentation. Index Structure - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///home/pfiguero/web_docs/
Search
Print
Home Bookmarks MandrakeSoft MandrakeStore MandrakeExpert MandrakeClub MandrakeC
file:///home/pfiguero/web_doc...nsive/papers_IndexDetail.html

Indexes

first
papers
space

InTml Documentation. Index Structure

(InTml home page)

papers **Index Identifier**

_unknown
- travel.MoveUnrestricted
- ualberta.apps.campusDemoInTs.Motorcycle
- ualberta.apps.campusDemoInTs.MotorcycleIT
- ualberta.selection.SelectByRayIT

barrilleaux01
- barrilleaux.ActivatebyTime
- barrilleaux.AddLabel
- barrilleaux.CreateLabel
- barrilleaux.DObject
- barrilleaux.DRM_MoveOffset
- barrilleaux.ObjectBarrilleaux
- barrilleaux.OrbitCamera

Figure 6.3: Category view of InTml.

- Filter details. It presents a general description of the filter, its interface – input and output ports –, its position in other categories, details about its ports, and bibliography. It is shown in the bottom right panel in both views (file and category). An example is shown in Figure 6.2.
- Composed filter details. It presents a general description of the technique, its interface, its implementation – object holders, filter instances –, details about ports, and bibliography.
- Application details. It presents a summary of the tasks – InTs and filters –, objects, and devices that the application uses, with a detailed information of connections and purposes of each element.

The program that generates this documentation performs the following tasks

- It creates the necessary directories and copies standard style files.
- In the case of the file view, it generates the list of files available in a predefined directory, and it creates all the necessary files with the detailed view for each filter class.
- Specialized views are generated for platforms and applications. A platform view shows input and output devices by separate, and an application view shows separate lists of objects, input devices, output devices, constants, object holders, filters and bindings. Both views require still more work and they are part of our list of future work

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- In the case of the category view, it extracts first the categories in the files. Once categories are found, it gets the classification for all classes in all available files. Initially, classes are classified in the category _unknown, and this initial value changes according to the real one, if any is assigned. Finally, files for each category are generated. Classes that do not define a value for a particular category are shown as _unknown, as it is shown in Figure 6.3. A designer can force a class to be left out of a category, by defining its category as _hidden. For example, the Button device is left out of the papers category since it does not apply to it.

Since there are several files in analysis and generation at the same time, the implementation uses some extensions to XSLT defined by the Xalan [4] implementation of XSLT. Some extra shell and AWK [40] scripts were used for the creation of the file and category indexes, since this task are either out of the scope of the XSLT processing power or they required several passes over several InTml files, which is slow in current XSLT implementations.

6.4.2 InTml Compiler

The InTml Compiler allows developers to transform InTml documents into executable code. Our current implementation reads InTml definitions, adds behavior code for filters, and creates source code in Java3D that represents the InTml application [8]. Figure 6.4 shows the general architecture of the implementation.

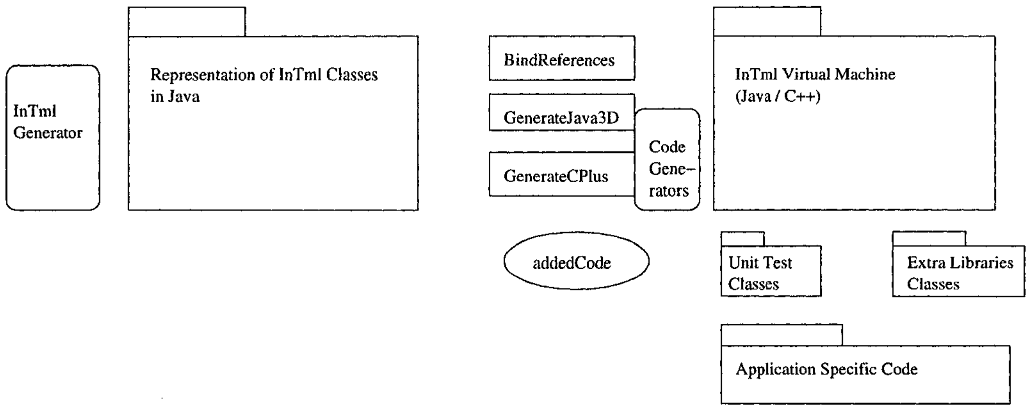


Figure 6.4: Architecture of the InTml Compiler. Rounded boxes represent code generators, rectangles are classes, ellipsis are files, and we use the UML symbol and semantics for a package of classes.

InTmlGenerator is a program generator written in TL [23] that creates a data structure in Java from the definitions in the InTml files. Such a structure is used for validation of the InTml files and for code generation. The compiler uses the Visitor pattern [44] in order to traverse the data structure and create the corresponding source code. This source code is a set of subclasses from the InTml virtual machine abstract classes, with added code for the particular behavior of each filter. The added code requires a special treatment:

1. Source code from InTml is created with special marks, where the programmer can add behavior–specific code.
2. Once the code is compiled, behavior–specific code is extracted from all files created by the compiler and it gets saved in a different file.

[8]There is an implementation for C++ under development.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. If the InTml files are compiled again, the new generated code will include the previous version of behavior–specific code, which was saved in the previous task.

This mechanism allowed us to develop at the same time both the behavior–specific code and the InTml virtual machine.

Let us follow an example of how the process works. Let us assume we want to implement the InTml class in Listing 35. The class has to be given with all other related classes and types, and with information about how to translate simple types into the underlying language, in this case Java.

Listing 35 InTml code for Generic3DOFTracker.

```
<DeviceClass id="Generic3DOFTracker">
  <ShortDesc>A generic orientation tracker</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="eulerAngles" type="Vector3">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
</DeviceClass>
```

InTmlGenerator takes all the InTml files and generate Java code that creates a graph–like structure in memory. This code has to be compiled and then traversed in order to create the Java files that represent the InTml class. Listing 36 shows the basic code generated for the class in Figure 35. The generator defines the class, its superclass, the fixed code for constructors, execution method, setup method (shared code among constructors), the code for creating ports, and the code for loading and saving properties (Properties are parameters that can be stored in a file and loaded at runtime, for configurability purposes).

The zones enclosed by // -=-=-=-=- are developer–specific code. The possible changes include changes in the superclass (in the case of specialized code not visible at the InTml level), the specific execution function, extra constructor code, changes in the accessor methods, changes in the default values for attributes, code for loading properties, and extra methods and attributes. This code is saved in a file in order to allow future changes in the InTml class without losing the actual code created by the developer. Listing 38 shows the structure of the extra code in this file. Each possible addition to the standard class is considered, and actual code is enclosed by `CDATA` statements. Developers should take into account the special requirements for an InTml filter class: it should not change the state of the scene, it should avoid dependencies not shown at the InTml level, and it should not modify the events it receive. This is of particular importance in the design of accessor methods and extra attributes and methods, since they can bypass the general restrictions of the InTml model.

6.4.3 InTml Checker

The purpose of the checker is to validate a set of InTml documents, and report semantic problems to the user. Currently, it is implemented as the first stage of the compiler. In its current state, the checker is capable of detecting the following basic problems:

- Referenced names not found, in applications and in composed filters. In particular, `Import`, `Implements`, or `Overrides` references not found.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 36 Initial Java code for Generic3DOFTracker.

```
public class Generic3DOFTracker
// -=-=-=-=-= Extends/Implements definitions here
  extends Device
// -=-=-=-=-= End Extends/Implements
{
  public void execute()
  {
      super.execute();
      // Do whatever is necessary with the information collected in iports.
      // Send information through the oports.
// -=-=-=-=-= Extra execute() code here
// -=-=-=-=-= End extra execute() code
  }
  public Generic3DOFTracker( ) {  super( );  }
  public void setup( String n )
  {
        super.setup( n );
// -=-=-=-=-= Extra constructor code here
// -=-=-=-=-= End extra constructor code
        createPorts();
  }
  private void createPorts()
  {
        // Add OPort of type javax.vecmath.Vector3f
        eulerAngles =  new AbsOPort( this, "eulerAngles" );
        oports.add( eulerAngles );
        // Add OPort of type javax.vecmath.Quat4f
        q =  new AbsOPort( this, "q" );
        oports.add( q );
  };
// -=-=-=-=-= Accessor methods here
// -=-=-=-=-= End accessor methods

  // Attributes
  AbsOPort eulerAngles;
  AbsOPort q;
// -=-=-=-=-= Extra methods and attributes here
// -=-=-=-=-= End extra methods and attributes

  public void saveDefaultProperties()
  {
      Properties p = new Properties();
      String filename = new String( "Properties" + File.separator +
                                    getClass().getName() + ".properties" );
// -=-=-=-=-= Default values for properties
// -=-=-=-=-= End default values for properties
      try
      {
        FileOutputStream f = new FileOutputStream( filename );
        p.store( f, "Properties for " + getClass().getName() );
      }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 37 Initial Java code for Generic3DOFTracker (Second part).

```
        catch( Exception e )
        {
          StdOut.getInstance().addMsg( 1, this, "Error saving properties :"
                                       + filename );
        }
    }
    public void loadProperties()
    {
        super.loadProperties();
        Properties p = new Properties();
        String filename = new String( "Properties" + File.separator +
                                      getClass().getName() + ".properties" );
        try
        {
          FileInputStream f = new FileInputStream( filename );
          p.load( f );
        }
        catch( Exception e )
        {
          StdOut.getInstance().addMsg( 1, this, "Properties file not found :"
                                       + filename );
        }
  // -=-=-=-=-= Loading properties
  // -=-=-=-=-= End loading properties
    }
  };
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 38 Developer–specific code for Generic3DOFTracker.

```
<Class id="Generic3DOFTracker">
  <AddedPackages>
  <![CDATA[  import javax.vecmath.*;
import javax.media.j3d.*;
import intmlRT.Loader.*;
]]>
  </AddedPackages>
  <Extensions>
  <![CDATA[  extends Device
]]>
  </Extensions>
  <InsideExecute>
  <![CDATA[  if (error == 0)
    {
      float[] e;
      e = this.getHMDValues();

      float yaw = (float)((e[0] + yawOffsetProperty.floatValue())* Math.PI/180);
      float pitch = (float)(e[1] * Math.PI/180);
      float roll = (float)(e[2] * Math.PI/180);
      Quat4f quat = computeb(pitch,yaw,roll);
      lastQ = quat;
      AbsInfo qinfo = new AbsInfo(quat);
      q.push(qinfo);
    }
    else
    {
       Quat4f quat = new Quat4f();
       AxisAngle4f aa = new AxisAngle4f(0.0f,1.0f,0.0f,0.0f);
       quat.set(aa);
       quat.normalize();
       AbsInfo qinfo = new AbsInfo(quat);
       q.push(qinfo);
    }
]]>
  </InsideExecute>
  <InsideConstructor>
  <![CDATA[  System.loadLibrary("HMD");
        error = this.startHMD();
        if (error == 1)
            {
                System.out.println("Error opening the HMD.");
            }
]]>
  </InsideConstructor>
  <NewAccessors>
  <![CDATA[  javax.vecmath.Quat4f getq( )
  {  return lastQ;
  }
]]>
  </NewAccessors>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Listing 39 Developer–specific code for Generic3DOFTracker (Second part).

```
    <AddedElements>
    <![CDATA[  Quat4f lastQ = new Quat4f();
      Quat4f computeb( float pitch, float yaw, float roll )
      {
          return compute(
                   -1 *((float) Math.PI*2f - roll),
                   -1 *((float) Math.PI*2f - yaw),
                   -1 * (pitch + (float)Math.PI) );
      }
      Quat4f compute( float pitch, float yaw, float roll )
      {
          Quat4f qu = new Quat4f();
          setQ( qu, pitch, yaw, roll );
          return qu;
      }
     void setQ( Quat4f q, float pitch, float yaw, float roll )
      {
          Quat4f qx = new Quat4f((float) Math.cos(pitch/2f),
                                 (float) Math.sin(pitch/2f), 0, 0);
          Quat4f qy = new Quat4f((float) Math.cos(yaw/2f), 0,
                                 (float) Math.sin(yaw/2f),0);
          Quat4f qz = new Quat4f((float) Math.cos(roll/2f), 0, 0,
                                 (float) Math.sin(roll/2f) );
          Quat4f qt = new Quat4f();
          qx.normalize(); qy.normalize(); qz.normalize();
          //System.out.println( "\tqx: " + "(" + qx + ")" );
          //System.out.println( "\tqy: " + "(" + qy + ")" );
          //System.out.println( "\tqz: " + "(" + qz + ")" );
          qt.set( qx );
          qt.mul( qy );
          qt.mul( qz );
          q.set( qt );
          q.normalize();
      }
      public native int startHMD();
      public native int stopHMD();
      public native float[] getHMDValues();
      int error;
      Integer comNumberProperty;
      Float yawOffsetProperty;
  ]]>
    </AddedElements>
    <DefaultProperties>
    <![CDATA[  p.setProperty("port","3");
  ]]>
    </DefaultProperties>
    <LoadProperties>
    <![CDATA[  String cString = p.getProperty("port","3");
        comNumberProperty = ConstantTranslator.String2java_lang_Integer(cString);
        System.out.println("PORT IS " + comNumberProperty);
        String yString = p.getProperty("yawOffset","-90.0");
        yawOffsetProperty = ConstantTranslator.String2java_lang_Float(yString);
        System.out.println("YAW OFFSET IS"+yawOffsetProperty);
  ]]>
    </LoadProperties>
  </Class>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Name convention errors. For example, an application's name can not include dots in order to avoid problems with conventions for package names, or subcomponents can not be referenced inside an object holder.
- Incorrect types in a binding. The type of events of the output and input ports connected in a binding elements must match.
- Input or output ports not found in a binding.
- Problems with overriding classes. An application can only override other applications, a filter class only other filter classes, and so on.
- Name conflicts between packages, filter classes, or instances.
- More than one `Implements` in a class [9]

The implementation of the checking task is an intrinsic part of the code for creating the data structure from the InTml files. It is implemented as `else` scenarios during the code, in which the error is registered, so it can be shown to the user at the end of the compilation stage. While this basic approach produced the desired effects, it is also intermingled with the code, which makes it difficult to maintain. Future versions of the checker will separate this code in order to make it more readable. In fact, some checking tasks are now described as part of visitors, which better enclose the processing task on top of the data structure.

6.5 InTml Framework Implementation

The InTml framework provides the semantics of the InTml specification over a particular core framework. Two implementations have been done, one in Java and one in C++. Both implementations contain the classes shown in Listing 40.

Listing 40 Classes in the InTml Framework. Indentation is used to show inheritance.

```
Filter
    InTmlSystem
    InT                 // Concrete class for filters
    Device
    GObject
    ObjectHolder
AbsApp
    TSApp
AbsPort
    IPort<T>            // C++ only
AbsOPort
    OPort<T>            // C++ only
AbsInfo
    Info<T>             // C++ only
```

All elements inside an application can be seen as filters, according to the inheritance hierarchy in the implementation. The current implementation of the execution model is embedded in the TSApp class, which makes a topological sort of the filters in the application, after breaking the cycles. There are many ways of breaking the cycles, so this may create different execution orders. In the Java implementation, all information through ports is considered of the same type, which may be too wide. In the C++ version, we used templates

[9]The current implementation supports only one `implements` clause.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in order to get a finer control of types, and we plan to explore the new implementation of templates in Java in order to have the same behavior.

The C++ implementation of the library tries to reproduce as close as possible the Java version, for maintenance purposes. There are two noticeable differences: ports are templates, so it is possible to have static type checking for the information flowing in the dataflow, and a basic garbage collection technique is added, since this facility was used in the Java implementation.

Code generation for Java is fully supported by the compiler. C++ components still have to create their code from scratch, but it is trivial to add a C++ generator to the current one for Java.

6.6 Library Implementation

The library of InTml elements is not implemented yet, and it is one of the main elements of future development. We plan to create also a test application for each element in the library, in order to provide a examples of use to designers. This collection will facilitate the understanding of the several options available in the library, and at the same time will provide an excellent method of testing.

6.7 Core Frameworks

As we have said, we have implemented InTml in two environments, one for Java and its set of libraries, and one in C++ and Performer. Other implementations are possible, but special care should be taken with the limitations that such frameworks might have. For example, a VRML implementation is possible provided that it is required to either consider just one event at any particular time (the basic execution model for VRML), or a drastic design that allows nodes to act with the semantic of InTml filters.

Each implementation presents its own advantages and difficulties: while garbage collection was useful for event handling in Java, the environment also made the implementation difficult with the non uniform implementation for common devices (mouse, keyboard) and others (trackers, for example). We had to use different methods to collect the information and provide a common interface, which makes the implementation a little bit awkward.

6.7.1 Runtime Environment

The runtime environment executes an InTml application over a platform. We developed three implementations of runtime environments, two in Java and one in C++.

The first implementation executes a Java implementation of an InTml application, which is found by reflection techniques. Such an implementation makes the runtime environment very simple, but it makes changes in InTml more difficult, since they have to be translated into a Java class and then compiled in order take effect.

The second implementation solved this problem and allowed a dynamic loading of all elements in an InTml application. Such files can be loaded from either the local file disk or from the Internet, which makes it very convenient for future web-based deployments. However, it makes the runtime environment dependent on the Internet, which fails in stand–alone computers, especially in presentations.

The third environment is a dynamic loader for C++, which runs in Linux and IRIX machines. This implementation provides an InTml solution in CAVE–like environments. Further testing is required to give more stability to such an environment.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.8 Future Work

In the area of tools, we have several plans for future developments. A visual programming environment will allow designers to create applications in an easier way, without direct contact with XML documents. This will make the structure of an application more evident, and it will allow the creation of more complex applications.

A library of InTml components has to be implemented, rich enough so that designers could create applications without developers aid. In this way, designers will have an environment for rapid prototyping of VR applications.

A tool for automatic retargeting is also in our plans. The main idea is to take advantage of a library of retargeted applications, and give suggestions to users of how a new application can be retargeted to previously known platforms. This will require a database of examples of retargeting, which will require further use of InTml.

Further developments in the InTml framework will provide a more complete coverage of the concepts in the InTml semantics. For example, composed filters and delays are still not supported in the InTml frameworks.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 7

Conclusions

We believe that Virtual Reality technology will be more important in the future, once the appropriate tools are available to a wider audience. We think our work helps to move the state of the art in this direction, and our results will help to provide a better support for designers of VR applications.

We have separated the concerns related to any VR application into two sets, one easy to understand without much technical background and the other highly technical. The former level talks about behavior, content, and devices at a high level of abstraction, in a way that facilitates the comprehension and design of complex VR applications. The latter level allows developers to divide their work in clearly identified components, in a way that facilitates development, reuse, and testing.

We define the concept of retargeting, as a new way for application portability, in which an application can change both devices and interaction techniques while at the same time reuse as much as possible of previous implementations. Retargeting provides a way for effective migration of VR applications, so actual rapid prototyping becomes possible in this field. This feature will facilitate the creation of several options, and as a result it will provide a better understanding of user's requirements and the required solution.

The formal description of the semantics of our proposal will allow several implementations of our framework, in very different ways. We believe this provides some freedom for improved implementations, once more data is available about this technique. It will also provide ways to test properties of the InTml environment at the level of the specification, so important theorems can be provided and proven.

The separation of concerns we have defined allow us to support two different communities, involved in any multimedia development: artists who know about the interface and the way to communicate certain information, and programmers who know about the technology and how to make it work. With the creation of a language in the middle, we provide both a communication channel and a separation mechanism for these two communities.

Supporting tools can be highly improved, with the addition of new tools and the enhancement of current ones. As we have mentioned before, a visual programming environment will allow designers to create applications in an easier way, without direct contact with XML documents. This will make the structure of an application more evident, and it will allow the creation of more complex applications. This area will probably require a fair amount of development, since high quality user interfaces are required in order to support a community of non–programmers. We will try to create proof–of–concept versions of some of the tools we believe are a priority, and offer them to the community for feedback purposes.

A library of InTml components have to be implemented, rich enough so that designers could create applications without developers aid. In this way, designers will have an environment for rapid prototyping of VR applications.

We also want to enhance the InTml language, in order to incorporate all concepts available in the formal specification. In this way, we will provide a more complete platform, and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a better starting point for the implementation of tools.

It is possible to further compare the systems for VR development presented in Chapter 2 with ours: Our system provides a clear and simple view of what a VR application is, better than the ones based on languages such as C++ or Java, and similar to X3D and Alice. Our separation of concerns is novel, although it can be applied in systems such as Alice, in which is possible to enrich the visual programming environment with newer devices and interaction techniques. Other systems such as HyNet [61], which uses Petri Nets for behavior specification, and PMIW [53], which uses a combination of dataflows plus state machine networks, also divide concerns and provide an abstract view of a VR application, but such views are more detailed than the one in InTml, and directed to developers of 3D interaction techniques.

Retargeting is also a novel concept in our thesis. It allows designers to change an application according to new requirements in a new platform, without dealing with specific details of several core technologies involved. InTml provides a clear structure for retargeting, which is not present in other systems available now. Our formal description provides a foundation for multiple implementations of InTml, which is necessary to provide real retargeting among diverse platforms. Despite other systems have a more complete user interface for developers (such as the ones for HyNet and PMIW), we believe we can provide a better environment, due to the fact that the concepts in InTml are easier to understand than the ones in such formalisms.

There are also several areas that we want to explore in the future. The separation of concerns also creates problems in understanding the overall implementation of the application, since the entire behavior is divided between high and low levels of abstraction. While we think this is not a problem for designers who do not want to know about specific implementation details, it may be a problem for developers who try to understand a piece of code during testing, since they also have to understand the InTml compilation process and the InTml framework. We want to do more work on this regard. There are also several ways to create an application in InTml, by applying different components and connections between them. We believe this may change in the future with the adoption of guidelines for InTml development, extracted from previous experiences. We also mentioned the following areas as future work in section 3.6:

- Actual comparative results between a parallel and a sequential implementation of InTml have to be performed. Ideally, we should be able to create some analytical methods for measuring performance advantages in a parallel implementation of a specific application, but we have to explore such possibility in more in detail.

- InTml is a new component technology that takes into account the specific quality attributtes required in VR applications. However, we require more applications implemented in this technology to valide more thoroughly its advantages and features. Such experimental test will require a community of users and an active developer community to support them.

- Exact time management and synchronization are issues not directly visible in InTml. This allows designers to worry about requirements without taking care of synchronization issues. However, such issues should be addressed in a succesful VR application. Our current approach forces designers to hide synchronization issues under filters that transparently handle several streams into a synchronized one. This approach has to be tested in more detail to understand its implications.

- More development tools and more complete ones are necessary in order to provide a true professional environment for VR designers.

- As InTml libraries grow, tools for finding and organizing them will be required, in order to make them usable and avoid work repetition.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Problems such as hardware calibration, hardware reliability, lack of 3D interaction standards, network issues, and storage issues are not addressed by our work, but we hope we provide a way to independently handle each one of these problems, in a way that is transparent to designers. This will allow the creation of better implementations with the same interface, which is the way other communities have grown and matured.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A

InTml Files for the Matching Application

We divided the InTml application in several files, according to the generality of the defined classes. The file `library.intml` comprises all general definitions, the ones that might be exported to a general purpose library. Such elements are:

- `Button`: It models when a button is clicked.
- `GenericJoystick`: Generic joystick definition
- `JXInputJoystick`: A joystick based on the JXInput interface
- `Generic3DOFTracker`: A generic orientation tracker
- `Generic6DOFTracker`: A generic position and orientation tracker
- `InterSenseWandTracker`: Hand tracker from InterSense
- `InterSenseHeadTracker`: Head tracker from InterSense
- `SpaceMouse`: Previously called Magellan 3D Controller.
- `GenericScreen`: A generic output screen
- `PCScreen`: A generic output screen from a PC
- `IGlasses`: Virtual i-glasses HMD
- `VirtualWindow`: Passive 3D Stereo display by Dimension Technologies
- `VisroomScreens`: Screens at the Visroom
- `GenericMouse`: A generic mouse definition
- `GenericKeyboard`: It defines the a keyboard as a collection of keys. It contains only some important keys right now... Because this limited implementation doesn't use arrays of ports, the description is rather long.
- `SMARTBoard`: It defines the important events from any SMART Board type of interface.
- `Timer`: Timer in the execution environment.
- `VRObject`: VRObjectClass defines the input and output ports of any VRObject in the dataflow. Transformations are organized as follows: Scales, Rotations, Translations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- `Viewpoint`: Viewpoint defines a basic element with just position and orientation. In the futute, it might include a description for the avatar.
- `Scene`: A Scene is a set of unstructured geometry that can't be selected, and a set of VRobjects, which can be selected. Its implementation defines the general structures required for selection computation.
- `ControlableFilter`: Defines the ports that a controlable filter should understand. It is basically a way to disable a filter.
- `Switch`: It doesn't generate repetitions due receiving the same signal several times.
- `SelectByRay`: Implements details of selection by intersection with a ray. It receives position updates of a ray in the scene and it computes which object from the scene is intersected first. Internally, it takes into account additions and removals of objects, to and from the scene.
- `SelectBy3DRay`: Implements details of selection by intersection with a ray, given a ray specification as 3D position plus orientation. It receives position and orientation updates of a ray in the scene and it computes which object from the scene is intersected first. Internally, it takes into account additions and removals of objects, to and from the scene.
- `SelectByTouching`: It selects an object that is colliding with the virtual pointer. It executes its operation everytime a signal is received by the 'compute' input port. This varies from previous Selection Techniques, and makes more clear that the position and orientation can be obtained from the 'handRepr'. Semantically is more correct, but it forces the system to wait two frames in order to make a selection since, as it was in previous selection techniques, selection used to receive the proposed changes in position and orientation, changes that were not commited at the time of execution of such filter.
- `GoGoSelection`: GoGo computes a new hand position as a function of the distance of the real hand from the chest, K, and D. It also computes a virtual position for the hand representation, as it is described in [71].
- `Joystick2Ray`: It computes a ray from the joystick position. It is useful for selection by a ray in a joystick-based environment. It uses refPos as the center of the plane that will contain posRay, and refQ as the normal of such plane.
- `HMDJ2Ray`: It computes a ray from the HMD and joystick positions, that can be used by a SelectByRay filter. The cursor is always centered in the field of view, with an offset given by the current joystick position. The ray is defined as centered between the user's eyes and passing through the joystick cursor (posRay).

The file `newClasses.intml` has application dependent definitions, those filters that are important for the behavior of this application. Such filters are:

- `RandomPQ`: It assigns a random new position to an object that is received in objs. If there are several objects in objs, it orders them by name and then assigns them a new position and orientation.
- `RandomRelativePQ`: Assigns a random position and orientation to an object within several constraints. Firstly, the xy of the position is fixed based on a grid which divides the screen into six sections so as to maximize screen usage and minimize occlusion. Secondly, the distance between each object and its copy must be equal for each pair of objects.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- `TransparentCopy`: It creates a new object with half the transparency of the previous one, and sends it through objCopied. It changes neither position not orientation, so the copy has to be moved later on to be distinguisable. If several objects are received, all are copied. The copy of an object with identifier "id" is called "CopyOfid".
- `HighlightedFeedback`: It changes the color of an object to a translucent green appearance. Used for selection feedback.
- `RotationBehavior`: It rotates an object received in the "obj" port, between the events buttonPressed and buttonReleased
- `TranslationBehavior`: It translates an object received in the "obj" port, between the events buttonPressed and buttonReleased. The events movXZ, movXY, movYZ change the plane of movement. By default, the movement is in plane XY
- `EchoPQ`: It echoes the position and orientation input events to the output. Its purpose is to move objects in 3D just when this function is activated.
- `RotTrans`: It manages both translation and rotation, from a pos3D and a quaternion as input. The object received at obj is moved when the buttonPressed signal is received. If any of the toggle signals is received, the object will be either just moved or just rotated. Movements are relative to the position and orientation of the pointer when the grab signal is received.
- `MatchFunction`: It measures the difference in orientation and position between an object and its copy.
- `DeleteWhenSignal`: It accumulates a set of objects and deletes them once a signal is received.
- `QuitMatching`: It waits until the three objects have been matched, sends a log event and ends the application two frames later. It also quits if the user presses a designated letter.
- `OrientationCenter`: It returns the intersection point of an imaginary line from the user's viewpoint and orientation and the near clipping plane. It assumes the clipping plane is big enough to be intersected, and that the clipping plane is at a distance d, when the quaternion is pointing to -z.
- `HMDJPointer`: The position of the pointer in a HMDJ environment is given by the point computed by OrientationCenter, and the current joystick position. The output is always the last value for the head's position plus the joystick offset (pos and pos3D). If the joystick is in dragging mode (given by buttonPressed and buttonReleased), pos is shifted by the amount given through shiftAmount.
- `ControlMatching`: Disables selection when rotation or translation are executing. When rotation starts, selection and translation are disabled. When translation starts, selection and rotation are disabled. When rotation ends, selection and translation are enabled. If the port 'allowBothTransRot' receives true, both orientation and translation might be active at the same time.
- `ResetPosition`: It allows to decouple a position from a device and a position from an object. When the signal is received, the last position is saved and all new positions generated are relative to the last one. It is useful when you want to come back to the origin.
- `Log`: It logs all the relevant information from an application. It also logs an execution counter, everytime it runs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The other files in the example define either basic characteristics of an application in a platform, or the matching application itself, retargeted to each platform. Such files are [1]:

- `generic3DDesktop.intml`: It defines elements present in all 3DDesktop application, which are a keyboard, a screen, a mouse3D, and an object holder for the viewpoint.
- `genericHMDJ.intml`: It defines a keyboard, a joystick, a 3DOF tracker, and an IGlasses HMD as standard devices in any HMDJ–based application; and a viewpoint, whose orientation is connected to the output of the 3DOF tracker.
- `genericPC.intml`: It defines a keyboard, a mouse, a screen, and a viewpoint as the basic elements in any PC–based application.
- `genericSMARTBoard.intml`: It defines a keyboard, a SMART Board and a viewpoint as the basic devices in any SMARTBoard–based application.
- `genericVisroom.intml`: It defines the basic devices in the Visroom — a wand, a head tracker, and the screens — plus the viewpoint as the basic elements in any Visroom–based application.
- `matching-3ddesktop.intml`: Matching application, 3DDesktop version. It connects the 3D mouse to a hand representation geometry, which is used to select object by colliding with them.
- `matching-hmdj.intml`: Matching application, HMDJ version. It uses the joystick and the head orientation to define the position of a pointer, which in turn allows selection by ray casting.
- `matching-pc.intml`: Matching application, PC version. It uses the standard keyboard and mouse to select and move objects.
- `matching-smartboard.intml`: Matching application, SMART Board version. It connects pens in the SMART Board to rotation and movement operations in the application.
- `matching-visroom.intml`: Matching application, Visroom version. It uses the Go Go interaction technique for selection and manipulation of objects [2].

The following sections are a copy of the files already described, and the code for each version of the application.

Library File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
        "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  library: Predefined classes, from the developer to the designer.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.
```

[1]We skip here some files for newer versions of the application and some test files.
[2]This application has not been implemented yet.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Changes:
May 8, 2002 : First version
May 23, 2002: PCScreen is created, in order to keep GenericScreen
              as an abstract superclass of all display devices.
  -->
<!-- Devices in the Matching Test application    -->
<Package id="matchingTest">

<DeviceClass id="Button">
  <ShortDesc>Basic Button</ShortDesc>
  <Description>
    It models when a button is clicked.
  </Description>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="clicked" type="boolean">
    <ShortDesc>true when it's been clicked (pressed and released)</ShortDesc>
  </OPort>
  <OPort id="pressed" type="boolean">
    <ShortDesc>true when it's been pressed </ShortDesc>
  </OPort>
  <OPort id="released" type="boolean">
    <ShortDesc>true when it's been released </ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="GenericJoystick">
  <ShortDesc>Generic joystick definition</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="xPos" type="int">
    <ShortDesc>Position left-right of the joystick</ShortDesc>
  </OPort>
  <OPort id="yPos" type="int">
    <ShortDesc>Position front-back of the joystick</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos2D">
    <ShortDesc>(xPos, yPos)</ShortDesc>
  </OPort>
  <OPort id="button1Pressed" type="AnyType">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button2Pressed" type="AnyType">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button3Pressed" type="AnyType">
    <ShortDesc>Button 3 pressed.</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </OPort>
  <OPort id="button4Pressed" type="AnyType">
    <ShortDesc>Button 4 pressed.</ShortDesc>
  </OPort>
  <OPort id="button1Released" type="AnyType">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button2Released" type="AnyType">
    <ShortDesc>Button 2 released.</ShortDesc>
  </OPort>
  <OPort id="button3Released" type="AnyType">
    <ShortDesc>Button 3 released.</ShortDesc>
  </OPort>
  <OPort id="button4Released" type="AnyType">
    <ShortDesc>Button 4 released.</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="JXInputJoystick">
  <ShortDesc>A joystick based on the JXInput interface</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="transform" type="Transform3D">
    <ShortDesc>The information from JXInput</ShortDesc>
  </OPort>
  <OPort id="button1Pressed" type="boolean">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button2Pressed" type="boolean">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button3Pressed" type="boolean">
    <ShortDesc>Button 3 pressed.</ShortDesc>
  </OPort>
  <OPort id="button1Released" type="boolean">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button2Released" type="boolean">
    <ShortDesc>Button 2 released.</ShortDesc>
  </OPort>
  <OPort id="button3Released" type="boolean">
    <ShortDesc>Button 3 released.</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="Generic3DOFTracker">
  <ShortDesc>A generic orientation tracker</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="eulerAngles" type="Vector3">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
</DeviceClass>

<DeviceClass id="Generic6DOFTracker">
  <ShortDesc>A generic position and orientation tracker</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <!-- Implements Generic3DOFTracker -->
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the tracker</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="InterSenseWandTracker">
  <ShortDesc>Hand tracker from InterSense</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <!-- Implements GenericJoystick -->
  <OPort id="xPos" type="int">
    <ShortDesc>Position left-right of the joystick</ShortDesc>
  </OPort>
  <OPort id="yPos" type="int">
    <ShortDesc>Position front-back of the joystick</ShortDesc>
  </OPort>
  <OPort id="posJoystick" type="Pos2D">
    <ShortDesc>(xPos, yPos)</ShortDesc>
  </OPort>
  <OPort id="button1Pressed" type="AnyType">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button2Pressed" type="AnyType">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button3Pressed" type="AnyType">
    <ShortDesc>Button 3 pressed.</ShortDesc>
  </OPort>
  <OPort id="button1Released" type="AnyType">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button2Released" type="AnyType">
    <ShortDesc>Button 2 released.</ShortDesc>
  </OPort>
  <OPort id="button3Released" type="AnyType">
    <ShortDesc>Button 3 released.</ShortDesc>
  </OPort>
  <!-- Implements Generic6DOFTracker -->
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the tracker</ShortDesc>
  </OPort>
  <OPort id="button4Pressed" type="AnyType">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <ShortDesc>Button 4 pressed.</ShortDesc>
  </OPort>
  <OPort id="button4Released" type="AnyType">
    <ShortDesc>Button 4 released.</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="InterSenseHeadTracker">
  <ShortDesc>Head tracker from InterSense</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <!-- Implements Generic6DOFTracker -->
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the tracker</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="SpaceMouse">
  <ShortDesc>A Logitech's Space Mouse</ShortDesc>
  <Description>
    Previously called Magellan 3D Controller.
  </Description>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="button1Pressed" type="AnyType">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button2Pressed" type="AnyType">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button3Pressed" type="AnyType">
    <ShortDesc>Button 3 pressed.</ShortDesc>
  </OPort>
  <OPort id="button4Pressed" type="AnyType">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button5Pressed" type="AnyType">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button6Pressed" type="AnyType">
    <ShortDesc>Button 3 pressed.</ShortDesc>
  </OPort>
  <OPort id="button7Pressed" type="AnyType">
    <ShortDesc>Button 1 pressed.</ShortDesc>
  </OPort>
  <OPort id="button8Pressed" type="AnyType">
    <ShortDesc>Button 2 pressed.</ShortDesc>
  </OPort>
  <OPort id="button1Released" type="AnyType">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button2Released" type="AnyType">
    <ShortDesc>Button 2 released.</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </OPort>
  <OPort id="button3Released" type="AnyType">
    <ShortDesc>Button 3 released.</ShortDesc>
  </OPort>
  <OPort id="button4Released" type="AnyType">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button5Released" type="AnyType">
    <ShortDesc>Button 2 released.</ShortDesc>
  </OPort>
  <OPort id="button6Released" type="AnyType">
    <ShortDesc>Button 3 released.</ShortDesc>
  </OPort>
  <OPort id="button7Released" type="AnyType">
    <ShortDesc>Button 1 released.</ShortDesc>
  </OPort>
  <OPort id="button8Released" type="AnyType">
    <ShortDesc>Button 2 released.</ShortDesc>
  </OPort>
  <!-- Implements Generic6DOFTracker -->
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the 3D mouse</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the 3D mouse</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="GenericScreen">
  <ShortDesc>A generic output screen</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Output"/>
  </Indexes>
</DeviceClass>

<DeviceClass id="PCScreen">
  <ShortDesc>A generic output screen from a PC</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="Devices.Output"/>
  </Indexes>
  <Implements classId="GenericScreen"/>
  <OPort id="nearClippingPlane" type="float">
    <ShortDesc>Near clipping plane</ShortDesc>
  </OPort>
  <OPort id="farClippingPlane" type="float">
    <ShortDesc>Far clipping plane</ShortDesc>
  </OPort>
  <OPort id="FOV" type="float">
    <ShortDesc>Field of View</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="IGlasses">
  <ShortDesc>Virtual i-glasses HMD</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="basic" value="intml.Devices.Output"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    </Indexes>
    <Implements classId="PCScreen"/>
  </DeviceClass>

  <DeviceClass id="VirtualWindow">
    <ShortDesc>Passive 3D Stereo display by Dimension Technologies</ShortDesc>
    <Description/>
    <Indexes>
      <Index id="basic" value="intml.Devices.Output"/>
    </Indexes>
    <Implements classId="PCScreen"/>
  </DeviceClass>

  <DeviceClass id="VisroomScreens">
    <ShortDesc>Screens at the Visroom</ShortDesc>
    <Description/>
    <Indexes>
      <Index id="basic" value="Devices.Output"/>
    </Indexes>
    <Implements classId="GenericScreen"/>
  </DeviceClass>

  <DeviceClass id="GenericMouse">
    <ShortDesc>A generic mouse definition </ShortDesc>
    <Description/>
    <Indexes>
      <Index id="basic" value="Devices.Input"/>
    </Indexes>
    <OPort id="xPos" type="int">
      <ShortDesc>Position in X.</ShortDesc>
    </OPort>
    <OPort id="yPos" type="int">
      <ShortDesc>Position in Y.</ShortDesc>
    </OPort>
    <OPort id="mousePos" type="Pos2D">
      <ShortDesc>Absolute position of the mouse pointer</ShortDesc>
      <Description>Adds up xPos and yPos into one output.</Description>
    </OPort>
    <OPort id="rButtonPressed" type="boolean">
      <ShortDesc>Right button pressed.</ShortDesc>
    </OPort>
    <OPort id="mButtonPressed" type="boolean">
      <ShortDesc>Middle button pressed.</ShortDesc>
    </OPort>
    <OPort id="lButtonPressed" type="boolean">
      <ShortDesc>Left button pressed.</ShortDesc>
    </OPort>
    <OPort id="rButtonReleased" type="boolean">
      <ShortDesc>Right button released.</ShortDesc>
    </OPort>
    <OPort id="mButtonReleased" type="boolean">
      <ShortDesc>Middle button released.</ShortDesc>
    </OPort>
    <OPort id="lButtonReleased" type="boolean">
      <ShortDesc>Left button released.</ShortDesc>
    </OPort>
  </DeviceClass>

  <DeviceClass id="GenericKeyboard">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <ShortDesc>A generic keyboard definition.</ShortDesc>
    <Description>
      It defines the a keyboard as a collection of keys. It contains only some
      important keys right now... Because this limited implementation doesn't
      use arrays of ports, the description is rather long.
    </Description>
    <Indexes>
      <Index id="basic" value="Devices.Input"/>
    </Indexes>
    <OPort id="a" type="boolean"><ShortDesc>Key 'a' pressed</ShortDesc></OPort>
    <OPort id="b" type="boolean"><ShortDesc>Key 'b' pressed</ShortDesc></OPort>
    <OPort id="c" type="boolean"><ShortDesc>Key 'c' pressed</ShortDesc></OPort>
    <OPort id="d" type="boolean"><ShortDesc>Key 'd' pressed</ShortDesc></OPort>
    <OPort id="q" type="boolean"><ShortDesc>Key 'q' pressed</ShortDesc></OPort>
    <OPort id="p" type="boolean"><ShortDesc>Key 'p' pressed</ShortDesc></OPort>
    <OPort id="s" type="boolean"><ShortDesc>Key 's' pressed</ShortDesc></OPort>
    <OPort id="x" type="boolean"><ShortDesc>Key 'x' pressed</ShortDesc></OPort>
    <OPort id="z" type="boolean"><ShortDesc>Key 'z' pressed</ShortDesc></OPort>
  </DeviceClass>

  <DeviceClass id="SMARTBoard">
    <ShortDesc>A SMART Board by SMART Technologies</ShortDesc>
    <Description>
      It defines the important events from any SMART Board type of interface.
    </Description>
    <Indexes>
      <Index id="basic" value="Devices.IO"/>
    </Indexes>
    <Implements classId="GenericScreen"/>
    <OPort id="xPos" type="int">
      <ShortDesc>Position in X.</ShortDesc>
    </OPort>
    <OPort id="yPos" type="int">
      <ShortDesc>Position in Y.</ShortDesc>
    </OPort>
    <OPort id="touchPos" type="Pos2D">
      <ShortDesc>Absolute position of a tool in the screen</ShortDesc>
      <Description>Adds up xPos and yPos into one output.</Description>
    </OPort>
    <OPort id="screenSelected" type="boolean">
      <ShortDesc>The screen has been touched with no pen selected.</ShortDesc>
    </OPort>
    <OPort id="screenReleased" type="boolean">
      <ShortDesc>The screen has been released with no pen selected.</ShortDesc>
    </OPort>
    <OPort id="pen1Selected" type="boolean">
      <ShortDesc>The pen 1 has been selected.</ShortDesc>
    </OPort>
    <OPort id="pen2Selected" type="boolean">
      <ShortDesc>The pen 2 has been selected.</ShortDesc>
    </OPort>
    <OPort id="pen3Selected" type="boolean">
      <ShortDesc>The pen 3 has been selected.</ShortDesc>
    </OPort>
    <OPort id="pen4Selected" type="boolean">
      <ShortDesc>The pen 4 has been selected.</ShortDesc>
    </OPort>
    <OPort id="pen1Released" type="boolean">
      <ShortDesc>The pen 1 has been released.</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </OPort>
  <OPort id="pen2Released" type="boolean">
    <ShortDesc>The pen 2 has been released.</ShortDesc>
  </OPort>
  <OPort id="pen3Released" type="boolean">
    <ShortDesc>The pen 3 has been released.</ShortDesc>
  </OPort>
  <OPort id="pen4Released" type="boolean">
    <ShortDesc>The pen 4 has been released.</ShortDesc>
  </OPort>
</DeviceClass>

<DeviceClass id="Timer">
  <ShortDesc>Timer in the execution environment</ShortDesc>
  <OPort id="secs" type="int">
    <ShortDesc>Seconds since the program initialization</ShortDesc>
  </OPort>
  <OPort id="curTime" type="Time">
    <ShortDesc>Seconds since the program initialization</ShortDesc>
  </OPort>
  <OPort id="ticks" type="int">
    <ShortDesc>Simulation steps</ShortDesc>
    <Description>Number of simulation steps, since the program initialization      </Description>
  </OPort>
</DeviceClass>

<ObjectClass id="VRObject">
  <ShortDesc>Interface for a VR object in the dataflow</ShortDesc>
  <Description>
VRObjectClass defines the input and output ports of any VRObject
in the dataflow.
        Transformations are organized as follows:
          Scales, Rotations, Translations
  </Description>
  <Indexes>
    <Index id="basic" value="Objects.VRObject"/>
  </Indexes>
  <IPort id="setPos" type="Pos3D">
    <ShortDesc>Changes the world position of an object</ShortDesc>
    <Description>
Changes the world-related position of an object.
It is considered a relative position if it is contained in
another object.
    </Description>
  </IPort>
  <OPort id="posChanged" type="Pos3D">
    <ShortDesc>Informs when the object moves</ShortDesc>
  </OPort>
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>Changes the world orientation of an object</ShortDesc>
    <Description>
Changes the world-related orientation of an object.
It is considered a relative orientation if it is contained in
another object.
    </Description>
  </IPort>
  <OPort id="qChanged" type="Quaternion">
    <ShortDesc>Informs when the object rotates</ShortDesc>
  </OPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <IPort id="setScale" type="Vector3">
    <ShortDesc>Changes the scale of an object</ShortDesc>
    <Description>
Changes the size of an object. It is considered a relative
size if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="scaleChanged" type="Vector3">
    <ShortDesc>Informs when the object changes its size</ShortDesc>
  </OPort>
  <IPort id="setMatrix" type="Matrix4">
    <ShortDesc>Changes the rigid transformations.</ShortDesc>
    <Description>
        Changes the rigid transformations that apply to the object.
It is useful when it necessary to apply rigid transformations
at once and at a specific order.
    </Description>
  </IPort>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds a new part to this object</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="VRObject">
    <ShortDesc>Informs when a part is added to the object</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Removes a part from this object</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="VRObject">
    <ShortDesc>Informs when a part is removed from the object</ShortDesc>
  </OPort>
  <IPort id="setTransparency" type="float">
    <ShortDesc>Defines the current transparency of an object (0-1)</ShortDesc>
  </IPort>
  <OPort id="transparencyChanged" type="float">
    <ShortDesc>Informs changes in the transparency</ShortDesc>
  </OPort>
  <IPort id="highlightState" type="boolean">
    <ShortDesc>true if the object has to appear higlighted</ShortDesc>
  </IPort>
  <OPort id="highlightChanged" type="boolean">
    <ShortDesc>Informs when the highlight state changes</ShortDesc>
  </OPort>
  <IPort id="setVisible" type="boolean">
    <ShortDesc>true if the object has to be visible</ShortDesc>
  </IPort>
  <OPort id="visibilityChanged" type="boolean">
    <ShortDesc>Informs when the visibility state changes</ShortDesc>
  </OPort>
</ObjectClass>

<ObjectClass id="Viewpoint">
  <ShortDesc>Representation of a viewpoint in the application</ShortDesc>
  <Description>
Viewpoint defines a basic element with just position and orientation.
In the futute, it might include a description for the avatar.
  </Description>
  <Indexes>
    <Index id="basic" value="Objects.Viewpoint"/>
  </Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <IPort id="setPos" type="Pos3D">
      <ShortDesc>Changes the world position of an object</ShortDesc>
      <Description>
Changes the world-related position of an object.
It is considered a relative position if it is contained in
another object.
      </Description>
    </IPort>
    <OPort id="posChanged" type="Pos3D">
      <ShortDesc>Informs when the object moves</ShortDesc>
    </OPort>
    <IPort id="setQ" type="Quaternion">
      <ShortDesc>Changes the world orientation of an object</ShortDesc>
      <Description>
Changes the world-related orientation of an object.
It is considered a relative orientation if it is contained in
another object.
      </Description>
    </IPort>
    <OPort id="qChanged" type="Quaternion">
      <ShortDesc>Informs when the object rotates</ShortDesc>
    </OPort>
    <IPort id="setEuler" type="Vector3">
      <ShortDesc>Changes orientation by new Euler angles</ShortDesc>
      <Description>
Changes the orientation of the viewpoint to the new
Euler angles.
      </Description>
    </IPort>
    <IPort id="setVisible" type="boolean">
      <ShortDesc>true if the object has to be visible</ShortDesc>
    </IPort>
    <OPort id="visibilityChanged" type="boolean">
      <ShortDesc>Informs when the visibility state changes</ShortDesc>
    </OPort>
</ObjectClass>

<ObjectClass id="Scene">
    <ShortDesc>Interface for a Scene in the dataflow</ShortDesc>
    <Description>
        A Scene is a set of unstructured geometry that can't be selected, and a
        set of VRobjects, which can be selected. Its implementation defines the
        general structures required for selection computation.
    </Description>
    <Indexes>
      <Index id="basic" value="Objects.Scene"/>
    </Indexes>
    <!-- Implements VRObject -->
    <IPort id="setPos" type="Pos3D">
      <ShortDesc>Changes the world position of an object</ShortDesc>
      <Description>
Changes the world-related position of an object.
It is considered a relative position if it is contained in
another object.
      </Description>
    </IPort>
    <OPort id="posChanged" type="Pos3D">
      <ShortDesc>Informs when the object moves</ShortDesc>
    </OPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>Changes the world orientation of an object</ShortDesc>
    <Description>
Changes the world-related orientation of an object.
It is considered a relative orientation if it is contained in
another object.
    </Description>
  </IPort>
  <OPort id="qChanged" type="Quaternion">
    <ShortDesc>Informs when the object rotates</ShortDesc>
  </OPort>
  <IPort id="setScale" type="Vector3">
    <ShortDesc>Changes the scale of an object</ShortDesc>
    <Description>
Changes the size of an object. It is considered a relative
size if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="scaleChanged" type="Vector3">
    <ShortDesc>Informs when the object changes its size</ShortDesc>
  </OPort>
  <IPort id="setMatrix" type="Matrix4">
    <ShortDesc>Changes the rigid transformations.</ShortDesc>
    <Description>
        Changes the rigid transformations that apply to the object.
It is useful when it necessary to apply rigid transformations
at once and at a specific order.
    </Description>
  </IPort>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds a new part to this object</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="VRObject">
    <ShortDesc>Informs when a part is added to the object</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Removes a part from this object</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="VRObject">
    <ShortDesc>Informs when a part is removed from the object</ShortDesc>
  </OPort>
  <IPort id="setTransparency" type="float">
    <ShortDesc>Defines the current transparency of an object (0-1)</ShortDesc>
  </IPort>
  <OPort id="transparencyChanged" type="float">
    <ShortDesc>Informs changes in the transparency</ShortDesc>
  </OPort>
  <IPort id="highlightState" type="boolean">
    <ShortDesc>true if the object has to appear higlighted</ShortDesc>
  </IPort>
  <OPort id="highlightChanged" type="boolean">
    <ShortDesc>Informs when the highlight state changes</ShortDesc>
  </OPort>
</ObjectClass>

<FilterClass id="ControlableFilter">
  <ShortDesc>Generic behavior of a filter</ShortDesc>
  <Description>
    Defines the ports that a controlable filter should understand.
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    It is basically a way to disable a filter.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="on" type="boolean">
      <ShortDesc>Defines if the filter is working or not</ShortDesc>
      <Description>
At startup, every filter is off. If an event with 'true' is
received, the filter is activated, and is the event's value is 'false'
the filter is deactivated, so no other input is considered.
If 'false' is received by this port,
the filter's state should be kept, so the next time it is turned on
it will continue its execution.
      </Description>
  </IPort>
  <IPort id="flushState" type="AnyType">
      <ShortDesc>Clears the current internal state</ShortDesc>
      <Description>
If anything is received in this port, the internal state of the
filter is set as if it were running for the first time.
      </Description>
  </IPort>
  <OPort id="inMode" type="AnyType">
      <ShortDesc>Event sent when the filter is executing</ShortDesc>
      <Description>
This port sends events everytime the filter is performing an
operation. It can be used for modal operations, when only one
has to be active at a particular time of the execution.
      </Description>
  </OPort>
  <OPort id="finishing" type="AnyType">
      <ShortDesc>Event sent when the filter is about to finish</ShortDesc>
      <Description>
This event is sent after the execution of the filter has ended,
and we want to tell other filters to react to this.
      </Description>
  </OPort>
</FilterClass>

<FilterClass id="Switch">
  <ShortDesc>Turns on and off, from two signals</ShortDesc>
  <Description>
    It doesn't generate repetitions due receiving the same signal
    several times.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="signalOn" type="AnyType">
<ShortDesc>On signal</ShortDesc>
  </IPort>
  <IPort id="signalOff" type="AnyType">
<ShortDesc>Off signal</ShortDesc>
  </IPort>
  <OPort id="onOff" type="boolean">
<ShortDesc>Boolean from the two inputs</ShortDesc>
  </OPort>
</FilterClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<FilterClass id="SelectByRay">
  <ShortDesc>Selection by intersection with a ray from the image plane</ShortDesc>
  <Description>
    Implements details of selection by intersection with a ray.
    It receives position updates of a ray in the scene
    and it computes which object from the scene is intersected first.
    Internally, it takes into account additions and removals of objects, to
    and from the scene.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="pos" type="Pos2D">
<ShortDesc>Position in the image plane for the selection</ShortDesc>
  </IPort>
  <IPort id="scene" type="Scene">
<ShortDesc>Selectable objects</ShortDesc>
  </IPort>
  <OPort id="object" type="VRObject">
<ShortDesc>Selected object</ShortDesc>
  </OPort>
  <OPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="SelectBy3DRay">
  <ShortDesc>Selection by intersection with a ray in 3D</ShortDesc>
  <Description>
    Implements details of selection by intersection with a ray, given a
    ray specification as 3D position plus orientation.
    It receives position and orientation updates of a ray in the scene
    and it computes which object from the scene is intersected first.
    Internally, it takes into account additions and removals of objects, to
    and from the scene.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="pos" type="Pos3D">
<ShortDesc>Position in the image plane for the selection</ShortDesc>
  </IPort>
  <IPort id="q" type="Quaternion">
<ShortDesc>Position in the image plane for the selection</ShortDesc>
  </IPort>
  <IPort id="scene" type="Scene">
<ShortDesc>Selectable objects</ShortDesc>
  </IPort>
  <OPort id="object" type="VRObject">
<ShortDesc>Selected object</ShortDesc>
  </OPort>
  <OPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </OPort>
</FilterClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<FilterClass id="SelectByTouching">
  <ShortDesc>Selection by collision with a virtual pointer.</ShortDesc>
  <Description>
      It selects an object that is colliding with the virtual pointer.
      It executes its operation everytime a signal is received by the
      'compute' input port. This varies from previous Selection Techniques,
      and makes more clear that the position and orientation can be
      obtained from the 'handRepr'. Semantically is more correct, but
      it forces the system to wait two frames in order to make a selection
      since, as it was in previous selection techniques, selection used to
      receive the proposed changes in position and orientation, changes that
      weren't commited at the time of execution of such filter.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="compute" type="AnyType" policy="ANY"></IPort>
  <IPort id="handRepr" type="VRObject"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <OPort id="object" type="VRObject"></OPort>
  <OPort id="deselected" type="AnyType">
<ShortDesc>Last selected object deselected</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="GoGoSelection">
  <ShortDesc>Go-Go Interaction Technique, by Poupyrev.</ShortDesc>
  <Description>
      GoGo computes a new hand position as a function of the distance
      of the real hand from the chest, K, and D. It also computes a
      virtual position for the hand representation, as it is described
      in the Poupyrev's paper.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="K" type="float" defValue="0.167"></IPort>
  <IPort id="D" type="float" defValue="0.6"></IPort>
  <IPort id="posHead" type="Pos3D"></IPort>
  <IPort id="qHead" type="Quaternion"></IPort>
  <IPort id="posHand" type="Pos3D"></IPort>
  <IPort id="qHand" type="Quaternion"></IPort>
  <IPort id="handRepr" type="VRObject"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <OPort id="object" type="VRObject"></OPort>
  <OPort id="gogoPos" type="Pos3D"></OPort>
  <OPort id="gogoQ" type="Quaternion"></OPort>
  <OPort id="deselected" type="AnyType">
<ShortDesc>Last selected object deselected</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="Joystick2Ray">
  <ShortDesc>Gets a ray from the joystick position</ShortDesc>
  <Description>
    It computes a ray from the joystick position. It is useful for selection
    by a ray in a joystick-based environment. It uses refPos
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    as the center of the plane that will contain posRay, and refQ as the
    normal of such plane.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <IPort id="pos" type="Pos2D">
<ShortDesc>Joystick's position</ShortDesc>
  </IPort>
  <IPort id="refPos" type="Pos3D" defValue="0, 0, 0">
<ShortDesc>Reference position</ShortDesc>
  </IPort>
  <IPort id="refQ" type="Quaternion" defValue="0, 0, 0, 1">
<ShortDesc>Reference orientation</ShortDesc>
  </IPort>
  <OPort id="posRay" type="Pos3D">
<ShortDesc>Position of the ray</ShortDesc>
  </OPort>
  <OPort id="qRay" type="Quaternion">
<ShortDesc>Rotation of the ray</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="HMDJ2Ray">
  <ShortDesc>Gets a ray from the HMD and joystick position</ShortDesc>
  <Description>
    It computes a ray from the HMD and joystick positions, that can be used
    by a SelectByRay filter. The cursor's is always centered in the field
    of view, with an offset given by the current joystick position. The
    ray is defined as centered between the user's eyes and passing through
    the joystick cursor (posRay).
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Selection"/>
  </Indexes>
  <IPort id="headQ" type="Quaternion">
<ShortDesc>Head orientation</ShortDesc>
  </IPort>
  <IPort id="headPos" type="Pos3D">
<ShortDesc>Head position</ShortDesc>
  </IPort>
  <IPort id="jPos" type="Pos2D">
<ShortDesc>Joystick position</ShortDesc>
  </IPort>
  <OPort id="posRay" type="Pos3D">
<ShortDesc>Position of the ray</ShortDesc>
  </OPort>
  <OPort id="qRay" type="Quaternion">
<ShortDesc>Rotation of the ray</ShortDesc>
  </OPort>
</FilterClass>

</Package>
```

Application Specific Classes

```
<?xml version="1.0" encoding="UTF-8"?>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
         "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  newClasses: Classes defined by the designer, used in the Matching test

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Changes:
May 8, 2002: First version
  -->
<!-- Devices in the Matching Test application    -->
<Package id="matchingTest">

<FilterClass id="RandomPQ">
  <ShortDesc>Assign a random position and orientation to an object</ShortDesc>
  <Description>
    It assigns a random new position to an object that is received in objs.
    If there are several objects in objs, it orders them by name and then
    assigns them a new position and orientation.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Init"/>
  </Indexes>
  <IPort id="objs" type="VRObject">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="gridDimension" type="int">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="Viewpoint">
    <ShortDesc>Current viewpoint</ShortDesc>
  </IPort>
  <IPort id="outputDevice" type="GenericScreen">
    <ShortDesc>It is used to know a valid new position</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="RandomRelativePQ">
  <ShortDesc>Assign a random position and orientation to pairs of objects</ShortDesc>
  <Description>
Assign a random position and orientation to an object within several
constraints.  Firstly, the xy of the position is fixed based on a grid
which divides the screen into six sections so as to maximize screen
usage and minimize occlusion.  Secondly, the distance between each
object and its copy must be equal for each pair of objects.
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Init"/>
  </Indexes>
  <IPort id="obj1" type="VRObject">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="copy1" type="VRObject">
    <ShortDesc>Object to be changed relative to the first object</ShortDesc>
  </IPort>
  <IPort id="obj2" type="VRObject">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="copy2" type="VRObject">
    <ShortDesc>Object to be changed relative to the first object</ShortDesc>
  </IPort>
  <IPort id="obj3" type="VRObject">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="copy3" type="VRObject">
    <ShortDesc>Object to be changed relative to the first object</ShortDesc>
  </IPort>
  <IPort id="outputDevice" type="GenericScreen">
    <ShortDesc>It is used to know a valid new position</ShortDesc>
  </IPort>
  <IPort id="viewPoint" type="Viewpoint">
    <ShortDesc>Used to know the position and orientation of viewpoint</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="TransparentCopy">
  <ShortDesc>Makes a transparent copy of an object</ShortDesc>
  <Description>
    It creates a new object with half the transparency of the previous one,
    and send it through objCopied. It doesn't change position nor orientation,
    so the copy has to be moved later on to be distinguisable. If several objects are
    received, all are copied. The copy of an object with identifier "id" is
    called "CopyOfid".
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Init"/>
  </Indexes>
  <IPort id="obj" type="VRObject" policy="ALL">
    <ShortDesc>Object to be copied</ShortDesc>
  </IPort>
  <OPort id="objCopied" type="VRObject">
    <ShortDesc>Transparent copy of the input</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="HighlightedFeedback">
  <ShortDesc>It changes an object's appearance to highlighted</ShortDesc>
  <Description>
    It changes the color of an object to a translucent green appearance.
    Used for selection feedback.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <IPort id="obj" type="VRObject">
<ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </IPort>
  <OPort id="currentObject" type="VRObject">
<ShortDesc>Current highlighted objects</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="RotationBehavior">
  <ShortDesc>It rotates an object when a button is dragged</ShortDesc>
  <Description>
    It rotates an object received in the "obj" port, between the events
    buttonPressed and buttonReleased
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="scaleMov" type="float">
    <ShortDesc>Scaling for the input movement</ShortDesc>
  </IPort>
  <IPort id="buttonPressed" type="boolean">
    <ShortDesc>Start the rotation of the object</ShortDesc>
  </IPort>
  <IPort id="buttonReleased" type="boolean">
    <ShortDesc>Stop the rotation of the object</ShortDesc>
  </IPort>
  <IPort id="pointerPos" type="Pos2D">
    <ShortDesc>Current pointer position</ShortDesc>
  </IPort>
  <IPort id="obj" type="VRObject">
    <ShortDesc>Object to be rotated</ShortDesc>
  </IPort>
  <IPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="TranslationBehavior">
  <ShortDesc>It translates an object when a button is dragged</ShortDesc>
  <Description>
    It translates an object received in the "obj" port, between the events
    buttonPressed and buttonReleased. The events movXZ, movXY, movYZ
    change the plane of movement. By default, the movement is in plane XY
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="scaleMov" type="float">
    <ShortDesc>Scaling for the input movement</ShortDesc>
  </IPort>
  <IPort id="buttonPressed" type="boolean">
    <ShortDesc>Start the rotation of the object</ShortDesc>
  </IPort>
  <IPort id="buttonReleased" type="boolean">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <ShortDesc>Stop the rotation of the object</ShortDesc>
  </IPort>
  <IPort id="pointerPos" type="Pos2D">
    <ShortDesc>Current pointer position</ShortDesc>
  </IPort>
  <IPort id="movXY" type="boolean">
    <ShortDesc>Changes the movement to the plane XY</ShortDesc>
  </IPort>
  <IPort id="movXZ" type="boolean">
    <ShortDesc>Changes the movement to the plane XZ</ShortDesc>
  </IPort>
  <IPort id="movYZ" type="boolean">
    <ShortDesc>Changes the movement to the plane YZ</ShortDesc>
  </IPort>
  <IPort id="obj" type="VRObject">
    <ShortDesc>Object to be rotated</ShortDesc>
  </IPort>
  <IPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="EchoPQ">
  <ShortDesc>Echoes position and orientation from input to output</ShortDesc>
  <Description>
    It echoes the position and orientation input events to the output. Its
    purpose is to move objects in 3D just when this function is activated.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Init"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="setPos" type="Pos3D">
    <ShortDesc>New position events</ShortDesc>
    <Description>
      Position events to be queued to the output
    </Description>
  </IPort>
  <OPort id="posChanged" type="Pos3D">
    <ShortDesc>Position events that have been received</ShortDesc>
  </OPort>
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>New orientation events</ShortDesc>
  </IPort>
  <OPort id="qChanged" type="Quaternion">
    <ShortDesc>Orientation events that have been received</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="RotTrans">
  <ShortDesc>Executes translation and rotation of an object</ShortDesc>
  <Description>
    It manages both translation and rotation, from a pos3D and a quaternion
    as input. the object received at obj is moved when the buttonPressed
    signal is received. If any of the toggle signals is received, the
    object will be either just moved or just rotated. Movements are relative
    to the position and orientation of the pointer when the grab signal is
    received.
  </Description>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="buttonPressed" type="boolean">
    <ShortDesc>Grab the object</ShortDesc>
  </IPort>
  <IPort id="buttonReleased" type="boolean">
    <ShortDesc>Release the object</ShortDesc>
  </IPort>
  <IPort id="toggleTranslation" type="AnyType">
    <ShortDesc>Turns on/off the translation behavior</ShortDesc>
  </IPort>
  <IPort id="toggleRotation" type="AnyType">
    <ShortDesc>Turns on/off the rotation behavior</ShortDesc>
  </IPort>
  <IPort id="obj" type="VRObject">
    <ShortDesc>Object to be rotated</ShortDesc>
  </IPort>
  <IPort id="deselected" type="boolean">
<ShortDesc>Last selected object deselected</ShortDesc>
  </IPort>
  <IPort id="setPos" type="Pos3D">
    <ShortDesc>New position events</ShortDesc>
    <Description>
      Position events to be queued to the output
    </Description>
  </IPort>
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>New orientation events</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="MatchFunction">
  <ShortDesc>It measures how far are two objects</ShortDesc>
  <Description>
    It measures the difference in orientation and position between an object
    and its copy.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
  <Implements classId="ControlableFilter"/>
  <IPort id="obj" type="VRObject">
<ShortDesc>Object that has been moved</ShortDesc>
  </IPort>
  <IPort id="copyObj" type="VRObject">
<ShortDesc>Reference object</ShortDesc>
  </IPort>
  <IPort id="compute" type="AnyType">
<ShortDesc>If info is received, this function is executed</ShortDesc>
  </IPort>
  <OPort id="match" type="VRObject">
<ShortDesc>The object is sent here when the two objects match</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="DeleteWhenSignal">
  <ShortDesc>Delete a set of objects when a signal is received</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Description>
    It accumulates a set of objects and deletes them once a signal is received.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Manipulation"/>
  </Indexes>
  <IPort id="obj" type="VRObject">
<ShortDesc>Objects that have to be deleted</ShortDesc>
  </IPort>
  <IPort id="signal" type="AnyType">
<ShortDesc>Signal that triggers the behavior</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="QuitMatching">
  <ShortDesc>Ends the application when the objects have been matched</ShortDesc>
  <Description>
    It waits until the three objects have been matched, send a log event
    and ends the application two frames later. It also quits if the user
    presses a designated letter.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="signal" type="AnyType">
<ShortDesc>Signal to quit the application</ShortDesc>
  </IPort>
  <IPort id="abortSignal" type="AnyType">
<ShortDesc>Signal to abort the application</ShortDesc>
  </IPort>
  <OPort id="endInfo" type="String">
<ShortDesc>Sent when the application is about the quit</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="OrientationCenter">
  <ShortDesc>Center of the current's head orientation</ShortDesc>
  <Description>
    It returns the intersection point of an imaginary line from the user's
    viewpoint and orientation and the near clipping plane. It assumes the
    clipping plane is big enough to be intersected, and that the clipping
    plane is at a distance d, when the quaternion is pointing to -z.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="p" type="Pos3D">
<ShortDesc>User's head position</ShortDesc>
  </IPort>
  <IPort id="q" type="Quaternion">
<ShortDesc>User's head orientation</ShortDesc>
  </IPort>
  <IPort id="screen" type="PCScreen">
<ShortDesc>Current screen parameters (i.e. clipping planes)</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>Intersection point in the clipping plane</ShortDesc>
  </OPort>
</FilterClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<FilterClass id="HMDJPointer">
  <ShortDesc>Defines the position of a pointer in a HMDJ</ShortDesc>
  <Description>
    The position of the pointer in a HMDJ environment is given by the
    point computed by OrientationCenter, and the current joystick
    position. The output is always the last value for the head's
    position plus the joystick offset (pos and pos3D). If the joystick is
    in dragging mode (given by buttonPressed and buttonReleased), pos is
    shifted by the amount given through shiftAmount.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="headCenter" type="Pos3D">
<ShortDesc>Intersection point in the clipping plane</ShortDesc>
  </IPort>
  <IPort id="jPos" type="Pos2D">
<ShortDesc>Joystick position, in the clipping plane</ShortDesc>
  </IPort>
  <IPort id="screen" type="PCScreen">
<ShortDesc>Current screen parameters (i.e. clipping planes)</ShortDesc>
  </IPort>
  <IPort id="shiftAmount" type="float">
<ShortDesc>Shift amount when dragging</ShortDesc>
  </IPort>
  <IPort id="buttonPressed" type="boolean">
    <ShortDesc>Start dragging mode for the joystick</ShortDesc>
  </IPort>
  <IPort id="buttonReleased" type="boolean">
    <ShortDesc>Stop dragging mode for the joystick</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos2D">
<ShortDesc>Pointer position over the enlarged clipping plane</ShortDesc>
  </OPort>
  <OPort id="pos3D" type="Pos3D">
<ShortDesc>3D Pointer position</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="ControlMatching">
  <ShortDesc>Control of InTs in execution</ShortDesc>
  <Description>
    Disables selection when rotation or translation are executing
    When rotation starts, selection and translation are disabled
    When translation starts, selection and rotation are disabled
    When rotation ends, selection and translation are enabled.
    If the port 'allowBothTransRot' receives true, both orientation
    and translation might be active at the same time.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="allowBothTransRot" type="boolean">
<ShortDesc>Allow both InTs active at the same time.</ShortDesc>
  </IPort>
  <IPort id="selection" type="ControlableFilter">
<ShortDesc>Selection technique</ShortDesc>
  </IPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <IPort id="rotation" type="ControlableFilter">
<ShortDesc>Rotation technique</ShortDesc>
  </IPort>
  <IPort id="translation" type="ControlableFilter">
<ShortDesc>Translation technique</ShortDesc>
  </IPort>
  <IPort id="inModeRotation" type="AnyType">
<ShortDesc>Rotation is executing</ShortDesc>
  </IPort>
  <IPort id="inModeTranslation" type="AnyType">
<ShortDesc>Translation is executing</ShortDesc>
  </IPort>
  <IPort id="endRotation" type="AnyType">
<ShortDesc>End of rotation</ShortDesc>
  </IPort>
  <IPort id="endTranslation" type="AnyType">
<ShortDesc>End of translation</ShortDesc>
  </IPort>
  <IPort id="match1" type="ControlableFilter">
<ShortDesc>Matching function 1</ShortDesc>
  </IPort>
  <IPort id="match2" type="ControlableFilter">
<ShortDesc>Matching function 2</ShortDesc>
  </IPort>
  <IPort id="match3" type="ControlableFilter">
<ShortDesc>Matching function 3</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="ResetPosition">
  <ShortDesc>Resets a position when a signal is received</ShortDesc>
  <Description>
    It allows to decouple a position from a device and a position from an
    object. When the signal is received, the last position is saved and
    all new positions generated are relative to the last one. It is useful
    when you want to come back to the origin.
  </Description>
  <Indexes>
    <Index id="basic" value="InTs.Init"/>
  </Indexes>
  <IPort id="setPos" type="Pos3D">
    <ShortDesc>new position</ShortDesc>
  </IPort>
  <IPort id="reset" type="AnyType">
    <ShortDesc>Set new zero</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>New position</ShortDesc>
  </OPort>
</FilterClass>


<FilterClass id="Log">
  <ShortDesc>Logs the application execution</ShortDesc>
  <Description>
    It logs all the relevant information from an application. It also logs
    an execution counter, everytime it runs.
  </Description>
  <Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <Index id="basic" value="InTs.Control"/>
  </Indexes>
  <IPort id="curTime" type="Time">
<ShortDesc>Current time</ShortDesc>
  </IPort>
  <IPort id="userId" type="String">
<ShortDesc>User id</ShortDesc>
  </IPort>
  <IPort id="matchSignal" type="VRObject">
<ShortDesc>Signal of a matching in the world</ShortDesc>
  </IPort>
  <IPort id="platformId" type="String">
<ShortDesc>Hardware platform identification</ShortDesc>
  </IPort>
  <IPort id="endSignal" type="AnyType">
<ShortDesc>Message before finishing the application</ShortDesc>
  </IPort>
  <IPort id="abortSignal" type="AnyType">
<ShortDesc>Message before aborting the application</ShortDesc>
  </IPort>
  <!-- Important events in the execution -->
  <IPort id="posObj1" type="Pos3D">
<ShortDesc>Position of object 1</ShortDesc>
  </IPort>
  <IPort id="posReplica1" type="Pos3D">
<ShortDesc>Position of replica of object 1</ShortDesc>
  </IPort>
  <IPort id="posObj2" type="Pos3D">
<ShortDesc>Position of object 2</ShortDesc>
  </IPort>
  <IPort id="posReplica2" type="Pos3D">
<ShortDesc>Position of replica of object 2</ShortDesc>
  </IPort>
  <IPort id="posObj3" type="Pos3D">
<ShortDesc>Position of object 3</ShortDesc>
  </IPort>
  <IPort id="posReplica3" type="Pos3D">
<ShortDesc>Position of replica of object 3</ShortDesc>
  </IPort>
  <IPort id="qObj1" type="Quaternion">
<ShortDesc>Orientation of object 1</ShortDesc>
  </IPort>
  <IPort id="qReplica1" type="Quaternion">
<ShortDesc>Orientation of replica of object 1</ShortDesc>
  </IPort>
  <IPort id="qObj2" type="Quaternion">
<ShortDesc>Orientation of object 2</ShortDesc>
  </IPort>
  <IPort id="qReplica2" type="Quaternion">
<ShortDesc>Orientation of replica of object 2</ShortDesc>
  </IPort>
  <IPort id="qObj3" type="Quaternion">
<ShortDesc>Orientation of object 3</ShortDesc>
  </IPort>
  <IPort id="qReplica3" type="Quaternion">
<ShortDesc>Orientation of replica of object 3</ShortDesc>
  </IPort>
  <IPort id="selectedObj" type="VRObject">
<ShortDesc>New selected Object</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </IPort>
  <IPort id="deselectedObj" type="AnyType">
<ShortDesc>Last object has been deselected</ShortDesc>
  </IPort>
  <IPort id="movXY" type="boolean">
    <ShortDesc>Changes the movement to the plane XY</ShortDesc>
  </IPort>
  <IPort id="movXZ" type="boolean">
    <ShortDesc>Changes the movement to the plane XZ</ShortDesc>
  </IPort>
  <IPort id="movYZ" type="boolean">
    <ShortDesc>Changes the movement to the plane YZ</ShortDesc>
  </IPort>
  <IPort id="pointerPos" type="Pos2D">
    <ShortDesc>Absolute position of the pointer</ShortDesc>
    <Description>Position of the pointer in the image plane.</Description>
  </IPort>
  <IPort id="rotateStart" type="boolean">
    <ShortDesc>Rotate functionality has been selected.</ShortDesc>
  </IPort>
  <IPort id="rotateStop" type="boolean">
    <ShortDesc>Rotate functionality has been released.</ShortDesc>
  </IPort>
  <IPort id="translateStart" type="boolean">
    <ShortDesc>Translate functionality has been selected.</ShortDesc>
  </IPort>
  <IPort id="translateStop" type="boolean">
    <ShortDesc>Translate functionality has been released.</ShortDesc>
  </IPort>
  <!-- 3D Mouse events -->
  <IPort id="toggleTranslation" type="AnyType">
    <ShortDesc>Turns on/off the translation behavior</ShortDesc>
  </IPort>
  <IPort id="toggleRotation" type="AnyType">
    <ShortDesc>Turns on/off the rotation behavior</ShortDesc>
  </IPort>

  <!-- Mouse events -->
  <!-- Keyboard events -->
  <!-- Joystick events -->
  <!-- Tracker events -->
</FilterClass>

</Package>
```

Matching Application. PC Version

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE App PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
          "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  matchingAppPC2: The matching application, standard PC version

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Assumptions
     - The initial viewpoint is looking to -z from (0,0,0)
     - The valid x, y positions are between (-20, 20)
  -->
<App id="matchingTest.matchingAppPC2">
  <ShortDesc>Matching application in the PC platform</ShortDesc>
  <Description>
    Rotation is doing by dragging the mouse with the left button
    Translation is doing by choosing a plane with a key (z/x/c) and dragging
      the mouse with the right button
    Random positions are now computed by RandomRelativePQ, which makes a
      better job than RandomPQ. Now random positions are deterministic, so
      we can define a position of all objects with just a number. The distance
      between an object and its copy is constant, and it also
      divides the screen in 6 areas, and put the objects in them.
  </Description>
  <Import id="matchingTest"/>
  <Overrides classId="matchingTest.genericPCApp"/>

  <!-- Load objects -->
  <Object id="obj1" filename="media/car.3ds" type="VRObject"/>
  <Object id="obj2" filename="media/Dodge32.3ds" type="VRObject"/>
  <Object id="obj3" filename="media/beethoven.obj" type="VRObject"/>
  <Object id="selectableObjs" type="Scene"/>
  <Binding iE="_self" iP="obj1" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj2" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj3" oE="selectableObjs" oP="addObject"/>

  <!-- Create viewpoint -->
  <Constant id="pV" type="Pos3D" value="0 0 0"/>
  <Constant id="qV" type="Quaternion" value="0 0 -1 0"/>
  <Constant id="notVisible" type="boolean" value="false"/>
  <Object id="viewpoint" filename="" type="Viewpoint"/>
  <Binding iE="_self" iP="pV" oE="viewpoint" oP="setPos"/>
  <Binding iE="_self" iP="qV" oE="viewpoint" oP="setQ"/>
  <Binding iE="_self" iP="notVisible" oE="viewpoint" oP="setVisible"/>
    <!-- Link it to the viewpoint in the system -->
  <Binding iE="_self" iP="viewpoint" oE="theCurrentViewpoint" oP="object"/>

  <!-- Localize objects at random -->
  <Constant id="dimGrid" type="int" value="10"/>
  <Filter id="randomRelativePQ" type="RandomRelativePQ"/>

  <Binding iE="_self" iP="obj1" oE="randomRelativePQ" oP="obj1"/>
  <Binding iE="_self" iP="obj2" oE="randomRelativePQ" oP="obj2"/>
  <Binding iE="_self" iP="obj3" oE="randomRelativePQ" oP="obj3"/>
    <!-- Bind the current output display to randomRelativePQ.outputDevice -->
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<Binding iE="_self" iP="screen" oE="randomRelativePQ" oP="outputDevice"/>

  <!-- Bind the current viewpoint to randomRelativePQ.outputDevice -->
<Binding iE="_self" iP="viewpoint" oE="randomRelativePQ" oP="viewPoint"/>

<!-- Create transparent copies of objects and localize them at random -->
<Filter id="tCopy1" type="TransparentCopy"/>
<Filter id="tCopy2" type="TransparentCopy"/>
<Filter id="tCopy3" type="TransparentCopy"/>
<Object id="transparentObjs" type="Scene"/>
<Binding iE="_self" iP="obj1" oE="tCopy1" oP="obj"/>
<Binding iE="_self" iP="obj2" oE="tCopy2" oP="obj"/>
<Binding iE="_self" iP="obj3" oE="tCopy3" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="randomRelativePQ" oP="copy1"/>
<Binding iE="tCopy2" iP="objCopied" oE="randomRelativePQ" oP="copy2"/>
<Binding iE="tCopy3" iP="objCopied" oE="randomRelativePQ" oP="copy3"/>
<Binding iE="tCopy1" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy2" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy3" iP="objCopied" oE="transparentObjs" oP="addObject"/>

<!-- Make selectableObjs selectable -->
  <!-- Create selection technique and bind it as necessary -->
<Filter id="selection" type="SelectByRay"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
<Binding iE="mouse" iP="mousePos" oE="selection" oP="pos"/>


<!-- Give feedback of selection -->
<Filter id="highlight" type="HighlightedFeedback"/>
<Binding iE="selection" iP="object" oE="highlight" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="highlight" oP="deselected"/>

<!-- Grab an object (See translate or rotate) -->

<!-- Rotate an object -->
<Filter id="rotateObj" type="RotationBehavior"/>
<Binding iE="mouse" iP="lButtonPressed" oE="rotateObj" oP="buttonPressed"/>
<Binding iE="mouse" iP="lButtonReleased" oE="rotateObj" oP="buttonReleased"/>
<Binding iE="mouse" iP="mousePos" oE="rotateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="rotateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotateObj" oP="deselected"/>

<!-- Translate an object -->
<Filter id="translateObj" type="TranslationBehavior"/>
<Binding iE="mouse" iP="rButtonPressed" oE="translateObj" oP="buttonPressed"/>
<Binding iE="mouse" iP="rButtonReleased" oE="translateObj" oP="buttonReleased"/>
<Binding iE="mouse" iP="mousePos" oE="translateObj" oP="pointerPos"/>
<Binding iE="keyboard" iP="z" oE="translateObj" oP="movXY"/>
<Binding iE="keyboard" iP="x" oE="translateObj" oP="movXZ"/>
<Binding iE="keyboard" iP="c" oE="translateObj" oP="movYZ"/>
<Binding iE="selection" iP="object" oE="translateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="translateObj" oP="deselected"/>

<!-- Release an object -->
<!--   (Done by rotate and translate) -->

<!-- Compute matching function -->
<Filter id="matchFunction1" type="MatchFunction"/>
<Filter id="matchFunction2" type="MatchFunction"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<Filter id="matchFunction3" type="MatchFunction"/>
<Binding iE="_self" iP="obj1" oE="matchFunction1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="matchFunction1" oP="copyObj"/>
<Binding iE="_self" iP="obj2" oE="matchFunction2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="matchFunction2" oP="copyObj"/>
<Binding iE="_self" iP="obj3" oE="matchFunction3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="matchFunction3" oP="copyObj"/>
<Binding iE="obj1" iP="posChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj1" iP="qChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj2" iP="posChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj2" iP="qChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj3" iP="posChanged" oE="matchFunction3" oP="compute"/>
<Binding iE="obj3" iP="qChanged" oE="matchFunction3" oP="compute"/>

<!-- Control all InTs -->
<Filter id="control" type="ControlMatching"/>
<Binding iE="_self" iP="selection" oE="control" oP="selection"/>
  <!-- Link platform-specific InTs -->
<Binding iE="_self" iP="rotateObj" oE="control" oP="rotation"/>
<Binding iE="rotateObj" iP="inMode" oE="control" oP="inModeRotation"/>
<Binding iE="_self" iP="translateObj" oE="control" oP="translation"/>
<Binding iE="translateObj" iP="inMode" oE="control" oP="inModeTranslation"/>
<Binding iE="rotateObj" iP="finishing" oE="control" oP="endRotation"/>
<Binding iE="translateObj" iP="finishing" oE="control" oP="endTranslation"/>
<Binding iE="_self" iP="matchFunction1" oE="control" oP="match1"/>
<Binding iE="_self" iP="matchFunction2" oE="control" oP="match2"/>
<Binding iE="_self" iP="matchFunction3" oE="control" oP="match3"/>

<!-- Delete objects once they match -->
<Filter id="deleteObjs1" type="DeleteWhenSignal"/>
<Filter id="deleteObjs2" type="DeleteWhenSignal"/>
<Filter id="deleteObjs3" type="DeleteWhenSignal"/>
<Binding iE="_self" iP="obj1" oE="deleteObjs1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="deleteObjs1" oP="obj"/>
<Binding iE="matchFunction1" iP="match" oE="deleteObjs1" oP="signal"/>
<Binding iE="_self" iP="obj2" oE="deleteObjs2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="deleteObjs2" oP="obj"/>
<Binding iE="matchFunction2" iP="match" oE="deleteObjs2" oP="signal"/>
<Binding iE="_self" iP="obj3" oE="deleteObjs3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="deleteObjs3" oP="obj"/>
<Binding iE="matchFunction3" iP="match" oE="deleteObjs3" oP="signal"/>

<!-- End of the application -->
<Filter id="quit" type="QuitMatching"/>
<Binding iE="matchFunction1" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction2" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction3" iP="match" oE="quit" oP="signal"/>
<Binding iE="keyboard" iP="q" oE="quit" oP="abortSignal"/>

<!-- Log start/stop times -->
<Filter id="log" type="Log"/>
<IDevice id="timer" type="Timer"/>
<Binding iE="timer" iP="curTime" oE="log" oP="curTime"/>
<Binding iE="quit" iP="endInfo" oE="log" oP="endSignal"/>

<!-- Identify user and platform -->
<!-- Change this constant to an id for a particular user -->
<Constant id="userId" type="String" value="Test user"/>
<Constant id="platformId" type="String" value="PC"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="_self" iP="userId" oE="log" oP="userId"/>
  <Binding iE="_self" iP="platformId" oE="log" oP="platformId"/>

  <!-- log the experience -->
  <!-- initial transformations, selected objects, position and orientation
       while moving, match times -->
  <Binding iE="matchFunction1" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction2" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction3" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="mouse" iP="mousePos" oE="log" oP="pointerPos"/>
  <Binding iE="mouse" iP="lButtonPressed" oE="log" oP="rotateStart"/>
  <Binding iE="mouse" iP="lButtonReleased" oE="log" oP="rotateStop"/>
  <Binding iE="mouse" iP="rButtonPressed" oE="log" oP="translateStart"/>
  <Binding iE="mouse" iP="rButtonReleased" oE="log" oP="translateStop"/>
  <Binding iE="keyboard" iP="z" oE="log" oP="movXY"/>
  <Binding iE="keyboard" iP="x" oE="log" oP="movXZ"/>
  <Binding iE="keyboard" iP="c" oE="log" oP="movYZ"/>
  <Binding iE="keyboard" iP="q" oE="log" oP="abortSignal"/>
  <Binding iE="selection" iP="object" oE="log" oP="selectedObj"/>
  <Binding iE="selection" iP="deselected" oE="log" oP="deselectedObj"/>

  <Binding iE="obj1" iP="posChanged" oE="log" oP="posObj1"/>
  <Binding iE="obj1" iP="qChanged" oE="log" oP="qObj1"/>
  <Binding iE="obj2" iP="posChanged" oE="log" oP="posObj2"/>
  <Binding iE="obj2" iP="qChanged" oE="log" oP="qObj2"/>
  <Binding iE="obj3" iP="posChanged" oE="log" oP="posObj3"/>
  <Binding iE="obj3" iP="qChanged" oE="log" oP="qObj3"/>

  <ObjectHolder id="copy1"/>
  <ObjectHolder id="copy2"/>
  <ObjectHolder id="copy3"/>
  <Binding iE="tCopy1" iP="objCopied" oE="copy1" oP="object"/>
  <Binding iE="tCopy2" iP="objCopied" oE="copy2" oP="object"/>
  <Binding iE="tCopy3" iP="objCopied" oE="copy3" oP="object"/>

  <Binding iE="copy1" iP="posChanged" oE="log" oP="posReplica1"/>
  <Binding iE="copy1" iP="qChanged" oE="log" oP="qReplica1"/>
  <Binding iE="copy2" iP="posChanged" oE="log" oP="posReplica2"/>
  <Binding iE="copy2" iP="qChanged" oE="log" oP="qReplica2"/>
  <Binding iE="copy3" iP="posChanged" oE="log" oP="posReplica3"/>
  <Binding iE="copy3" iP="qChanged" oE="log" oP="qReplica3"/>

</App>
```

Matching Application. SB Version

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE App PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
        "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  matchingAppSMARTBoard: The matching application, SMART Board version

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Assumptions
    - The initial viewpoint is looking to -z from (0,0,0)
  -->
<App id="matchingTest.matchingAppSMARTBoard2">
  <ShortDesc>Matching application in the SMART Board platform</ShortDesc>
  <Import id="matchingTest"/>
  <Overrides classId="matchingTest.genericSMARTBoardApp"/>

  <!-- Load objects -->
  <Object id="obj1" filename="media/car.3ds" type="VRObject"/>
  <Object id="obj2" filename="media/Dodge32.3ds" type="VRObject"/>
  <Object id="obj3" filename="media/beethoven.obj" type="VRObject"/>
  <Object id="selectableObjs" type="Scene"/>
  <Binding iE="_self" iP="obj1" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj2" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj3" oE="selectableObjs" oP="addObject"/>

  <!-- Create viewpoint -->
  <Constant id="pV" type="Pos3D" value="0 0 0"/>
  <Constant id="qV" type="Quaternion" value="0 0 -1 0"/>
  <Constant id="notVisible" type="boolean" value="false"/>
  <Object id="viewpoint" filename="" type="Viewpoint"/>
  <Binding iE="_self" iP="pV" oE="viewpoint" oP="setPos"/>
  <Binding iE="_self" iP="qV" oE="viewpoint" oP="setQ"/>
  <Binding iE="_self" iP="notVisible" oE="viewpoint" oP="setVisible"/>
    <!-- Link it to the viewpoint in the system -->
  <Binding iE="_self" iP="viewpoint" oE="theCurrentViewpoint" oP="object"/>

  <!-- Localize objects at random -->
  <Filter id="randomRelativePQ" type="RandomRelativePQ"/>

  <Binding iE="_self" iP="obj1" oE="randomRelativePQ" oP="obj1"/>
  <Binding iE="_self" iP="obj2" oE="randomRelativePQ" oP="obj2"/>
  <Binding iE="_self" iP="obj3" oE="randomRelativePQ" oP="obj3"/>
    <!-- Bind the current output display to randomRelativePQ.outputDevice -->
  <Binding iE="_self" iP="smartboard" oE="randomRelativePQ" oP="outputDevice"/>

    <!-- Bind the current viewpoint to randomRelativePQ.outputDevice -->
  <Binding iE="_self" iP="viewpoint" oE="randomRelativePQ" oP="viewPoint"/>

  <!-- Create transparent copies of objects and localize them at random -->
  <Filter id="tCopy1" type="TransparentCopy"/>
  <Filter id="tCopy2" type="TransparentCopy"/>
  <Filter id="tCopy3" type="TransparentCopy"/>
  <Object id="transparentObjs" type="Scene"/>
  <Binding iE="_self" iP="obj1" oE="tCopy1" oP="obj"/>
  <Binding iE="_self" iP="obj2" oE="tCopy2" oP="obj"/>
  <Binding iE="_self" iP="obj3" oE="tCopy3" oP="obj"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<Binding iE="tCopy1" iP="objCopied" oE="randomRelativePQ" oP="copy1"/>
<Binding iE="tCopy2" iP="objCopied" oE="randomRelativePQ" oP="copy2"/>
<Binding iE="tCopy3" iP="objCopied" oE="randomRelativePQ" oP="copy3"/>
<Binding iE="tCopy1" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy2" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy3" iP="objCopied" oE="transparentObjs" oP="addObject"/>

<!-- Make selectableObjs selectable -->
<Filter id="selection" type="SelectByRay"/>
<Binding iE="smartboard" iP="touchPos" oE="selection" oP="pos"/>
<Binding iE="smartboard" iP="screenReleased" oE="selection" oP="flushState"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>

<!-- Give feedback of selection -->
<Filter id="highlight" type="HighlightedFeedback"/>
<Binding iE="selection" iP="object" oE="highlight" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="highlight" oP="deselected"/>
  <!-- Release the current state once a pen release the screen-->
<Binding iE="smartboard" iP="pen1Released" oE="selection" oP="flushState"/>
<Binding iE="smartboard" iP="pen2Released" oE="selection" oP="flushState"/>
<Binding iE="smartboard" iP="pen3Released" oE="selection" oP="flushState"/>
<Binding iE="smartboard" iP="pen4Released" oE="selection" oP="flushState"/>

<!-- Grab an object (See translate or rotate) -->

<!-- Rotate an object -->
<Filter id="rotateObj" type="RotationBehavior"/>
<Binding iE="smartboard" iP="pen1Selected" oE="rotateObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen1Released" oE="rotateObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="touchPos" oE="rotateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="rotateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotateObj" oP="deselected"/>

<!-- Translate an object -->
<Filter id="translateObj" type="TranslationBehavior"/>
<Binding iE="smartboard" iP="pen2Selected" oE="translateObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen2Released" oE="translateObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="pen2Selected" oE="translateObj" oP="movXY"/>
<Binding iE="smartboard" iP="pen3Selected" oE="translateObj" oP="movXZ"/>
<Binding iE="smartboard" iP="pen3Selected" oE="translateObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen3Released" oE="translateObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="pen4Selected" oE="translateObj" oP="movYZ"/>
<Binding iE="smartboard" iP="pen4Selected" oE="translateObj" oP="buttonPressed"/>
<Binding iE="smartboard" iP="pen4Released" oE="translateObj" oP="buttonReleased"/>
<Binding iE="smartboard" iP="touchPos" oE="translateObj" oP="pointerPos"/>
<Binding iE="selection" iP="object" oE="translateObj" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="translateObj" oP="deselected"/>

<!-- Release an object -->
<!--   (Done by rotate and translate) -->

<!-- Compute matching function -->
<Filter id="matchFunction1" type="MatchFunction"/>
<Filter id="matchFunction2" type="MatchFunction"/>
<Filter id="matchFunction3" type="MatchFunction"/>
<Binding iE="_self" iP="obj1" oE="matchFunction1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="matchFunction1" oP="copyObj"/>
<Binding iE="_self" iP="obj2" oE="matchFunction2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="matchFunction2" oP="copyObj"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<Binding iE="_self" iP="obj3" oE="matchFunction3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="matchFunction3" oP="copyObj"/>
<Binding iE="obj1" iP="posChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj1" iP="qChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj2" iP="posChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj2" iP="qChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj3" iP="posChanged" oE="matchFunction3" oP="compute"/>
<Binding iE="obj3" iP="qChanged" oE="matchFunction3" oP="compute"/>

<!-- Control all InTs -->
<Filter id="control" type="ControlMatching"/>
<Binding iE="_self" iP="selection" oE="control" oP="selection"/>
  <!-- Link platform-specific InTs -->
<Binding iE="_self" iP="rotateObj" oE="control" oP="rotation"/>
<Binding iE="rotateObj" iP="inMode" oE="control" oP="inModeRotation"/>
<Binding iE="_self" iP="translateObj" oE="control" oP="translation"/>
<Binding iE="translateObj" iP="inMode" oE="control" oP="inModeTranslation"/>
<Binding iE="rotateObj" iP="finishing" oE="control" oP="endRotation"/>
<Binding iE="translateObj" iP="finishing" oE="control" oP="endTranslation"/>
<Binding iE="_self" iP="matchFunction1" oE="control" oP="match1"/>
<Binding iE="_self" iP="matchFunction2" oE="control" oP="match2"/>
<Binding iE="_self" iP="matchFunction3" oE="control" oP="match3"/>

<!-- Delete objects once they match -->
<Filter id="deleteObjs1" type="DeleteWhenSignal"/>
<Filter id="deleteObjs2" type="DeleteWhenSignal"/>
<Filter id="deleteObjs3" type="DeleteWhenSignal"/>
<Binding iE="_self" iP="obj1" oE="deleteObjs1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="deleteObjs1" oP="obj"/>
<Binding iE="matchFunction1" iP="match" oE="deleteObjs1" oP="signal"/>
<Binding iE="_self" iP="obj2" oE="deleteObjs2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="deleteObjs2" oP="obj"/>
<Binding iE="matchFunction2" iP="match" oE="deleteObjs2" oP="signal"/>
<Binding iE="_self" iP="obj3" oE="deleteObjs3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="deleteObjs3" oP="obj"/>
<Binding iE="matchFunction3" iP="match" oE="deleteObjs3" oP="signal"/>

<!-- End of the application -->
<Filter id="quit" type="QuitMatching"/>
<Binding iE="matchFunction1" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction2" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction3" iP="match" oE="quit" oP="signal"/>
<Binding iE="keyboard" iP="q" oE="quit" oP="abortSignal"/>

<!-- Log start/stop times -->
<Filter id="log" type="Log"/>
<IDevice id="timer" type="Timer"/>
<Binding iE="timer" iP="curTime" oE="log" oP="curTime"/>
<Binding iE="quit" iP="endInfo" oE="log" oP="endSignal"/>

<!-- Identify user and platform -->
<!-- Change this constant to an id for a particular user -->
<Constant id="userId" type="String" value="Test user"/>
<Constant id="platformId" type="String" value="SMART Board"/>
<Binding iE="_self" iP="userId" oE="log" oP="userId"/>
<Binding iE="_self" iP="platformId" oE="log" oP="platformId"/>

<!-- log the experience -->
<!-- initial transformations, selected objects, position and orientation while
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      moving, match times -->
  <Binding iE="matchFunction1" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction2" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction3" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="smartboard" iP="touchPos" oE="log" oP="pointerPos"/>
  <Binding iE="smartboard" iP="pen1Selected" oE="log" oP="rotateStart"/>
  <Binding iE="smartboard" iP="pen1Released" oE="log" oP="rotateStop"/>
  <Binding iE="smartboard" iP="pen1Selected" oE="log" oP="translateStart"/>
  <Binding iE="smartboard" iP="pen1Released" oE="log" oP="translateStop"/>
  <Binding iE="smartboard" iP="pen2Selected" oE="log" oP="translateStart"/>
  <Binding iE="smartboard" iP="pen2Released" oE="log" oP="translateStop"/>
  <Binding iE="smartboard" iP="pen3Selected" oE="log" oP="translateStart"/>
  <Binding iE="smartboard" iP="pen3Released" oE="log" oP="translateStop"/>
  <Binding iE="smartboard" iP="pen4Selected" oE="log" oP="translateStart"/>
  <Binding iE="smartboard" iP="pen4Released" oE="log" oP="translateStop"/>
  <Binding iE="smartboard" iP="pen2Selected" oE="log" oP="movXY"/>
  <Binding iE="smartboard" iP="pen3Selected" oE="log" oP="movXZ"/>
  <Binding iE="smartboard" iP="pen4Selected" oE="log" oP="movYZ"/>
  <Binding iE="selection" iP="object" oE="log" oP="selectedObj"/>
  <Binding iE="selection" iP="deselected" oE="log" oP="deselectedObj"/>
  <Binding iE="keyboard" iP="q" oE="log" oP="abortSignal"/>

  <Binding iE="obj1" iP="posChanged" oE="log" oP="posObj1"/>
  <Binding iE="obj1" iP="qChanged" oE="log" oP="qObj1"/>
  <Binding iE="obj2" iP="posChanged" oE="log" oP="posObj2"/>
  <Binding iE="obj2" iP="qChanged" oE="log" oP="qObj2"/>
  <Binding iE="obj3" iP="posChanged" oE="log" oP="posObj3"/>
  <Binding iE="obj3" iP="qChanged" oE="log" oP="qObj3"/>

  <ObjectHolder id="copy1"/>
  <ObjectHolder id="copy2"/>
  <ObjectHolder id="copy3"/>
  <Binding iE="tCopy1" iP="objCopied" oE="copy1" oP="object"/>
  <Binding iE="tCopy2" iP="objCopied" oE="copy2" oP="object"/>
  <Binding iE="tCopy3" iP="objCopied" oE="copy3" oP="object"/>

  <Binding iE="copy1" iP="posChanged" oE="log" oP="posReplica1"/>
  <Binding iE="copy1" iP="qChanged" oE="log" oP="qReplica1"/>
  <Binding iE="copy2" iP="posChanged" oE="log" oP="posReplica2"/>
  <Binding iE="copy2" iP="qChanged" oE="log" oP="qReplica2"/>
  <Binding iE="copy3" iP="posChanged" oE="log" oP="posReplica3"/>
  <Binding iE="copy3" iP="qChanged" oE="log" oP="qReplica3"/>

</App>
```

Matching Application. HMDJ Version

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE App PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
          "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  hmdj: An environment with the I-glasses and a joystick

  It has the following fixed characteristics:
  - A joystick, a keyboard, and the i-glasses HMD, with its tracker
  - theRenderedScene is an ObjectHolder that contains the rendered scene
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  - theCurrentViewpoint is an ObjectHolder with the current viewpoint
  - the viewpoint rotates according to the tracker position

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<App id="matchingTest.hmdjApp">
  <ShortDesc>Generic app for an I-glasses HMD and a Joystick</ShortDesc>
  <Import id="matchingTest"/>
  <IDevice id="keyboard" type="GenericKeyboard"/>
  <IDevice id="joystick" type="GenericJoystick"/>
  <IDevice id="tracker" type="Generic3DOFTracker"/>
  <ODevice id="hmd" type="IGlasses"/>
  <ObjectHolder id="theCurrentViewpoint"/>
  <ObjectHolder id="theRenderedScene"/>

  <!-- Standard head behavior -->
  <Binding iE="tracker" iP="q" oE="theCurrentViewpoint" oP="setQ"/>

</App>
```

Matching Application. 3DD Version

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE App PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
          "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  matchingApp3DDesktop2: The matching application, 3D Desktop. Version 2

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  -->
<!-- Assumptions
     - The initial viewpoint is looking to -z from (0,0,0)
     - The valid x, y positions are between (-20, 20)
  -->
<App id="matchingTest.matchingApp3DDesktop3">
  <ShortDesc>Matching application in the 3D Desktop platform</ShortDesc>
  <Description>
    It uses the keyboard instead of the buttons in the 3dmouse.
    It uses RotTrans instead of EchoPQ
    Random positions are now computed by RandomRelativePQ, which makes a
      better job than RandomPQ. Now random positions are deterministic, so
      we can define a position of all objects with just a number. The distance
      between an object and its copy is constant, and it also
      divides the screen in 6 areas, and put the objects in them.
    It also has a key for reseting the pointer's position
  </Description>
  <Import id="matchingTest"/>
  <Overrides classId="matchingTest.generic3DDesktopApp"/>

  <!-- Load objects -->
  <Object id="obj1" filename="media/car.3ds" type="VRObject"/>
  <Object id="obj2" filename="media/Dodge32.3ds" type="VRObject"/>
  <Object id="obj3" filename="media/beethoven.obj" type="VRObject"/>
  <Object id="selectableObjs" type="Scene"/>
  <Binding iE="_self" iP="obj1" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj2" oE="selectableObjs" oP="addObject"/>
  <Binding iE="_self" iP="obj3" oE="selectableObjs" oP="addObject"/>

  <!-- Create viewpoint -->
  <Constant id="pV" type="Pos3D" value="0 0 0"/>
  <Constant id="qV" type="Quaternion" value="0 0 -1 0"/>
  <Constant id="notVisible" type="boolean" value="false"/>
  <Object id="viewpoint" filename="" type="Viewpoint"/>
  <Binding iE="_self" iP="pV" oE="viewpoint" oP="setPos"/>
  <Binding iE="_self" iP="qV" oE="viewpoint" oP="setQ"/>
  <Binding iE="_self" iP="notVisible" oE="viewpoint" oP="setVisible"/>
    <!-- Link it to the viewpoint in the system -->
  <Binding iE="_self" iP="viewpoint" oE="theCurrentViewpoint" oP="object"/>

  <!-- Localize objects at random -->
  <Filter id="randomRelativePQ" type="RandomRelativePQ"/>

  <Binding iE="_self" iP="obj1" oE="randomRelativePQ" oP="obj1"/>
  <Binding iE="_self" iP="obj2" oE="randomRelativePQ" oP="obj2"/>
  <Binding iE="_self" iP="obj3" oE="randomRelativePQ" oP="obj3"/>
    <!-- Bind the current output display to randomRelativePQ.outputDevice -->
  <Binding iE="_self" iP="screen" oE="randomRelativePQ" oP="outputDevice"/>

    <!-- Bind the current viewpoint to randomRelativePQ.outputDevice -->
  <Binding iE="_self" iP="viewpoint" oE="randomRelativePQ" oP="viewPoint"/>

  <!-- Create transparent copies of objects and localize them at random -->
  <Filter id="tCopy1" type="TransparentCopy"/>
  <Filter id="tCopy2" type="TransparentCopy"/>
  <Filter id="tCopy3" type="TransparentCopy"/>
  <Object id="transparentObjs" type="Scene"/>
  <Binding iE="_self" iP="obj1" oE="tCopy1" oP="obj"/>
  <Binding iE="_self" iP="obj2" oE="tCopy2" oP="obj"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<Binding iE="_self" iP="obj3" oE="tCopy3" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="randomRelativePQ" oP="copy1"/>
<Binding iE="tCopy2" iP="objCopied" oE="randomRelativePQ" oP="copy2"/>
<Binding iE="tCopy3" iP="objCopied" oE="randomRelativePQ" oP="copy3"/>
<Binding iE="tCopy1" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy2" iP="objCopied" oE="transparentObjs" oP="addObject"/>
<Binding iE="tCopy3" iP="objCopied" oE="transparentObjs" oP="addObject"/>

<!-- Make selectableObjs selectable -->
  <!-- Create selection technique and bind it as necessary -->
<Filter id="selection" type="SelectByTouching"/>
<Object id="vHand" filename="media/pointer.obj" type="VRObject"/>
<Binding iE="_self" iP="selectableObjs" oE="selection" oP="scene"/>
<Binding iE="mouse3D" iP="pos" oE="selection" oP="compute"/>
<Binding iE="mouse3D" iP="q" oE="selection" oP="compute"/>
<Binding iE="_self" iP="vHand" oE="selection" oP="handRepr"/>
<Filter id="resetPosition" type="ResetPosition"/>
<Binding iE="mouse3D" iP="pos" oE="resetPosition" oP="setPos"/>
<Binding iE="resetPosition" iP="pos" oE="vHand" oP="setPos"/>
<Binding iE="mouse3D" iP="q" oE="vHand" oP="setQ"/>

<!-- Give feedback of selection -->
<Filter id="highlight" type="HighlightedFeedback"/>
<Binding iE="selection" iP="object" oE="highlight" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="highlight" oP="deselected"/>

<!-- Grab an object (See translate or rotate) -->

<!-- Rotate and Translate an object -->
<Filter id="rotTrans" type="RotTrans"/>
<Binding iE="selection" iP="object" oE="rotTrans" oP="obj"/>
<Binding iE="selection" iP="deselected" oE="rotTrans" oP="deselected"/>
<Binding iE="keyboard" iP="z" oE="rotTrans" oP="buttonPressed"/>
<Binding iE="keyboard" iP="x" oE="rotTrans" oP="buttonReleased"/>
<Binding iE="keyboard" iP="a" oE="rotTrans" oP="toggleTranslation"/>
<Binding iE="keyboard" iP="s" oE="rotTrans" oP="toggleRotation"/>
<Binding iE="mouse3D" iP="q" oE="rotTrans" oP="setQ"/>
<Binding iE="mouse3D" iP="pos" oE="rotTrans" oP="setPos"/>

<!-- Release an object -->
<!--   (Done by rotate and translate) -->

<!-- Compute matching function -->
<Filter id="matchFunction1" type="MatchFunction"/>
<Filter id="matchFunction2" type="MatchFunction"/>
<Filter id="matchFunction3" type="MatchFunction"/>
<Binding iE="_self" iP="obj1" oE="matchFunction1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="matchFunction1" oP="copyObj"/>
<Binding iE="_self" iP="obj2" oE="matchFunction2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="matchFunction2" oP="copyObj"/>
<Binding iE="_self" iP="obj3" oE="matchFunction3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="matchFunction3" oP="copyObj"/>
<Binding iE="obj1" iP="posChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj1" iP="qChanged" oE="matchFunction1" oP="compute"/>
<Binding iE="obj2" iP="posChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj2" iP="qChanged" oE="matchFunction2" oP="compute"/>
<Binding iE="obj3" iP="posChanged" oE="matchFunction3" oP="compute"/>
<Binding iE="obj3" iP="qChanged" oE="matchFunction3" oP="compute"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<!-- Control all InTs -->
<Filter id="control" type="ControlMatching"/>
<Filter id="switch1" type="Switch"/>
<Filter id="switch2" type="Switch"/>
<Constant id="bothActive" type="boolean" value="true"/>
<Constant id="startFalse" type="boolean" value="false"/>
<Binding iE="_self" iP="bothActive" oE="control" oP="allowBothTransRot"/>
<Binding iE="_self" iP="selection" oE="control" oP="selection"/>
  <!-- Link platform-specific InTs -->
<Binding iE="_self" iP="rotTrans" oE="control" oP="rotation"/>
<Binding iE="_self" iP="rotTrans" oE="control" oP="translation"/>
<!-- Can both InTs be active at the same time? -->
<Binding iE="rotTrans" iP="inMode" oE="control" oP="inModeRotation"/>
<Binding iE="keyboard" iP="x" oE="control" oP="endRotation"/>
<Binding iE="rotTrans" iP="inMode" oE="control" oP="inModeTranslation"/>
<Binding iE="keyboard" iP="x" oE="control" oP="endTranslation"/>
<Binding iE="keyboard" iP="z" oE="switch1" oP="signalOn"/>
<Binding iE="keyboard" iP="x" oE="switch1" oP="signalOff"/>
<Binding iE="switch1" iP="onOff" oE="rotTrans" oP="on"/>
<Binding iE="keyboard" iP="z" oE="switch2" oP="signalOff"/>
<Binding iE="keyboard" iP="x" oE="switch2" oP="signalOn"/>
<Binding iE="switch2" iP="onOff" oE="selection" oP="on"/>
<Binding iE="_self" iP="matchFunction1" oE="control" oP="match1"/>
<Binding iE="_self" iP="matchFunction2" oE="control" oP="match2"/>
<Binding iE="_self" iP="matchFunction3" oE="control" oP="match3"/>
<Binding iE="matchFunction1" iP="match" oE="control" oP="endTranslation"/>
<Binding iE="matchFunction2" iP="match" oE="control" oP="endTranslation"/>
<Binding iE="matchFunction3" iP="match" oE="control" oP="endTranslation"/>
<Binding iE="matchFunction1" iP="match" oE="control" oP="endRotation"/>
<Binding iE="matchFunction2" iP="match" oE="control" oP="endRotation"/>
<Binding iE="matchFunction3" iP="match" oE="control" oP="endRotation"/>
<Binding iE="keyboard" iP="p" oE="resetPosition" oP="reset"/>


<!-- Delete objects once they match -->
<Filter id="deleteObjs1" type="DeleteWhenSignal"/>
<Filter id="deleteObjs2" type="DeleteWhenSignal"/>
<Filter id="deleteObjs3" type="DeleteWhenSignal"/>
<Binding iE="_self" iP="obj1" oE="deleteObjs1" oP="obj"/>
<Binding iE="tCopy1" iP="objCopied" oE="deleteObjs1" oP="obj"/>
<Binding iE="matchFunction1" iP="match" oE="deleteObjs1" oP="signal"/>
<Binding iE="_self" iP="obj2" oE="deleteObjs2" oP="obj"/>
<Binding iE="tCopy2" iP="objCopied" oE="deleteObjs2" oP="obj"/>
<Binding iE="matchFunction2" iP="match" oE="deleteObjs2" oP="signal"/>
<Binding iE="_self" iP="obj3" oE="deleteObjs3" oP="obj"/>
<Binding iE="tCopy3" iP="objCopied" oE="deleteObjs3" oP="obj"/>
<Binding iE="matchFunction3" iP="match" oE="deleteObjs3" oP="signal"/>

<!-- End of the application -->
<Filter id="quit" type="QuitMatching"/>
<Binding iE="matchFunction1" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction2" iP="match" oE="quit" oP="signal"/>
<Binding iE="matchFunction3" iP="match" oE="quit" oP="signal"/>
<Binding iE="keyboard" iP="q" oE="quit" oP="abortSignal"/>

<!-- Log start/stop times -->
<Filter id="log" type="Log"/>
<IDevice id="timer" type="Timer"/>
<Binding iE="timer" iP="curTime" oE="log" oP="curTime"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="quit" iP="endInfo" oE="log" oP="endSignal"/>

  <!-- Identify user and platform -->
  <!-- Change this constant to an id for a particular user -->
  <Constant id="userId" type="String" value="Test user"/>
  <Constant id="platformId" type="String" value="3DDesktop"/>
  <Binding iE="_self" iP="userId" oE="log" oP="userId"/>
  <Binding iE="_self" iP="platformId" oE="log" oP="platformId"/>

  <!-- log the experience -->
  <!-- initial transformations, selected objects, position and orientation
       while moving, match times -->
  <Binding iE="matchFunction1" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction2" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="matchFunction3" iP="match" oE="log" oP="matchSignal"/>
  <Binding iE="keyboard" iP="z" oE="log" oP="rotateStart"/>
  <Binding iE="keyboard" iP="x" oE="log" oP="rotateStop"/>
  <Binding iE="keyboard" iP="z" oE="log" oP="translateStart"/>
  <Binding iE="keyboard" iP="x" oE="log" oP="translateStop"/>
  <Binding iE="keyboard" iP="q" oE="log" oP="abortSignal"/>
  <Binding iE="selection" iP="object" oE="log" oP="selectedObj"/>
  <Binding iE="selection" iP="deselected" oE="log" oP="deselectedObj"/>

  <Binding iE="keyboard" iP="a" oE="log" oP="toggleTranslation"/>
  <Binding iE="keyboard" iP="s" oE="log" oP="toggleRotation"/>

  <Binding iE="obj1" iP="posChanged" oE="log" oP="posObj1"/>
  <Binding iE="obj1" iP="qChanged" oE="log" oP="qObj1"/>
  <Binding iE="obj2" iP="posChanged" oE="log" oP="posObj2"/>
  <Binding iE="obj2" iP="qChanged" oE="log" oP="qObj2"/>
  <Binding iE="obj3" iP="posChanged" oE="log" oP="posObj3"/>
  <Binding iE="obj3" iP="qChanged" oE="log" oP="qObj3"/>

  <ObjectHolder id="copy1"/>
  <ObjectHolder id="copy2"/>
  <ObjectHolder id="copy3"/>
  <Binding iE="tCopy1" iP="objCopied" oE="copy1" oP="object"/>
  <Binding iE="tCopy2" iP="objCopied" oE="copy2" oP="object"/>
  <Binding iE="tCopy3" iP="objCopied" oE="copy3" oP="object"/>

  <Binding iE="copy1" iP="posChanged" oE="log" oP="posReplica1"/>
  <Binding iE="copy1" iP="qChanged" oE="log" oP="qReplica1"/>
  <Binding iE="copy2" iP="posChanged" oE="log" oP="posReplica2"/>
  <Binding iE="copy2" iP="qChanged" oE="log" oP="qReplica2"/>
  <Binding iE="copy3" iP="posChanged" oE="log" oP="posReplica3"/>
  <Binding iE="copy3" iP="qChanged" oE="log" oP="qReplica3"/>

</App>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix B

InTml Files for the Virtual Clay Application

The InTml files for the virtual clay application are listed in this appendix. The file `library.intml` is the same one as in the matching application, and for this reason is omitted here.

Application Specific Classes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
        "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  newClasses: Classes defined by the designer, used in the Virtual Clay app.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Changes:
Aug 13, 2002: First version
  -->
<!-- Devices in the Matching Test application    -->
<Package id="virtualClay">
<Import id="library"/>
<Import id="uofaDevices"/>

<ObjectClass id="Material">
  <ShortDesc>Defines a deformable object with a certain density</ShortDesc>
  <Description>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Material models any material that can be deformed by or deform to
any other object.
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <IPort id="setPos" type="Pos3Df">
    <ShortDesc>Changes the world position of an object</ShortDesc>
    <Description>
Changes the world-related position of an object.
It is considered a relative position if it is contained in
another object.
    </Description>
  </IPort>
  <OPort id="currentPos" type="Pos3Df">
    <ShortDesc>Informs when the object moves</ShortDesc>
  </OPort>
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>Changes the world orientation of an object</ShortDesc>
    <Description>
Changes the world-related orientation of an object.
It is considered a relative orientation if it is contained in
another object.
    </Description>
  </IPort>
  <OPort id="currentQ" type="Quaternion">
    <ShortDesc>Informs when the object rotates</ShortDesc>
  </OPort>
  <IPort id="forces" type="Force">
    <ShortDesc>Current forces applied to this material</ShortDesc>
  </IPort>
  <IPort id="density" type="float">
    <ShortDesc>Density of the material, 1 for hard, 0 for vapor</ShortDesc>
  </IPort>
  <IPort id="deformedBy" type="Material">
    <ShortDesc>Object in contact</ShortDesc>
  </IPort>
  <IPort id="spin" type="SpinParameters">
    <ShortDesc>Makes the material spin over an axis</ShortDesc>
  </IPort>
  <IPort id="cutBy" type="Material">
    <ShortDesc>Cuts this object with the one received here</ShortDesc>
  </IPort>
  <OPort id="deformed" type="Deformation">
    <ShortDesc>Information about the deformation in the object</ShortDesc>
  </OPort>
  <OPort id="currentVel" type="Speed">
    <ShortDesc>Current velocity of the object</ShortDesc>
  </OPort>
  <OPort id="currentAccel" type="Acceleration">
    <ShortDesc>Current acceleration of the object</ShortDesc>
  </OPort>
  <OPort id="currentBSphere" type="VRObject">
    <ShortDesc>Current bounding sphere</ShortDesc>
  </OPort>
</ObjectClass>

<ObjectClass id="VirtualClayEnvironment">
  <ShortDesc>Environment for deformable objects</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Description>
The environment that coordinates deformation among objects
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <IPort id="addObject" type="Material">
    <ShortDesc>Adds a new part to this object</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="Material">
    <ShortDesc>Informs when a part is added to the object</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="Material">
    <ShortDesc>Removes a part from this object</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="Material">
    <ShortDesc>Informs when a part is removed from the object</ShortDesc>
  </OPort>
  <IPort id="splitObject" type="String">
    <ShortDesc>Splits an object into its pieces</ShortDesc>
  </IPort>
  <OPort id="splittedObject" type="String">
    <ShortDesc>Splitted object</ShortDesc>
  </OPort>
  <IPort id="compute" type="AnyType">
    <ShortDesc>Event that triggers a computation step</ShortDesc>
  </IPort>
  <OPort id="collidingObjects" type="Material2">
    <ShortDesc>Objects in a collision</ShortDesc>
    <Description>
      Objects involved in a collision, at a certain time. Right now handles
      collisions of two objects only, and only of of them will receive the
      'deformedBy' event (the one with lower density, or at random, if
      densities are the same). Material2 is an array of 2 objects of type
      Material
    </Description>
  </OPort>
</ObjectClass>

<FilterClass id="Mouse2Ray">
  <ShortDesc>Computer a ray in 3D from the mouse position</ShortDesc>
  <Description>
The mouse position defines a ray in the current screen
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <IPort id="pos2D" type="Pos2Di">
    <ShortDesc>Pointer position</ShortDesc>
  </IPort>
  <IPort id="screen" type="GenericScreen">
    <ShortDesc>Screen information</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="Viewpoint">
    <ShortDesc>Current viewpoint</ShortDesc>
  </IPort>
  <OPort id="ray" type="Ray">
    <ShortDesc>Current ray</ShortDesc>
  </OPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
</FilterClass>

<FilterClass id="IntersectByBSphere">
  <ShortDesc>Intersects an object by using its bounding sphere</ShortDesc>
  <Description>
The intersection is with a ray and gives a point
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <IPort id="ray" type="Ray">
    <ShortDesc>Intersecting ray</ShortDesc>
  </IPort>
  <IPort id="object" type="Material">
    <ShortDesc>Object to intersect</ShortDesc>
  </IPort>
  <IPort id="compute" type="AnyType">
    <ShortDesc>Causes a computation</ShortDesc>
  </IPort>
  <OPort id="point" type="Pos3Df">
    <ShortDesc>Point of intersection in the sphere</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="RotateByDragging">
  <ShortDesc>Rotates by using a point in its bounding sphere</ShortDesc>
  <Description>
The point gives the dragging behavior. The initial point is the
reference for the rotation. Subsequent points define the angle
of rotation. The initial point can be flushed by flushing the state
of the filter, available by the input ports from ControlableFilter
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <Implements id="ControlableFilter"/>
  <IPort id="object" type="Material">
    <ShortDesc>Object to rotate</ShortDesc>
  </IPort>
  <IPort id="point" type="Pos3Df">
    <ShortDesc>Point of the bounding sphere, for dragging</ShortDesc>
  </IPort>
</FilterClass>

<FilterClass id="TranslatePlane">
  <ShortDesc>Moves an object in a plane, given a 2D pos</ShortDesc>
  <Description>
The input position may come from a pointer, as a mouse or
a joystick
  </Description>
  <Indexes>
    <Index id="basic" value="Objects"/>
  </Indexes>
  <Implements id="ControlableFilter"/>
  <IPort id="object" type="Material">
    <ShortDesc>Object to move</ShortDesc>
  </IPort>
  <IPort id="pos2D" type="Pos2Di">
    <ShortDesc>Pointer position</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </IPort>
  <IPort id="plane" type="CartesianPlaneID">
    <ShortDesc>Plane of movement</ShortDesc>
  </IPort>
  <IPort id="scaleFactor" type="float">
    <ShortDesc>Scale factor for the pointer position</ShortDesc>
  </IPort>
</FilterClass>

</Package>
```

Application Specific Devices

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
         "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  Physical Devices: Translations and extensions to basic types in the
                    library.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- Changes:
Aun 13, 2002 : First version
  -->
<!-- Devices in the Matching Test application    -->
<Package id="uofaDevices">
<Import id="library"/>

<DeviceClass id="SpaceBall">
  <ShortDesc>A 3D Connexion's SpaceBall</ShortDesc>
  <Description>
    A 6DOF device, with a ball instead of the puck in the SpaceMouse
  </Description>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <OPort id="buttons" type="Button" isArray="true"
 typeArray="static" maxArray="12">
    <ShortDesc>SpaceBall buttons</ShortDesc>
  </OPort>
  <Device id="trackedPos" type="Generic6DOFTracker"/>
</DeviceClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<DeviceClass id="Phantom">
  <ShortDesc>A Phantom device</ShortDesc>
  <Description>
    A force-feedback arm
  </Description>
  <Indexes>
    <Index id="basic" value="Devices.Input"/>
  </Indexes>
  <IPort id="force" type="Force">
    <ShortDesc>Sets a new force in the device</ShortDesc>
  </IPort>
  <IPort id="torque" type="Torque">
    <ShortDesc>Sets a new torque in the device</ShortDesc>
  </IPort>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds a new object to collide with</ShortDesc>
  </IPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Removes an object from the collision list</ShortDesc>
  </IPort>
  <!-- Implements Generic6DOFTracker -->
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the tracker</ShortDesc>
  </OPort>
</DeviceClass>



</Package>
```

Virtual Clay Application. PC Version

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE App PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
         "http://www.cs.ualberta.ca/~pfiguero/InTmlTemp/spec/intml.dtd">
<!-- Copyright info
  virtualClayApp: The Virtual Clay application

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  -->
<App id="virtualClayPC.app">
  <ShortDesc>Virtual Clay application. PC Version</ShortDesc>
  <Import id="library"/>
  <Import id="uofaDevices"/>

  <IDevice id="keyboard" type="GenericKeyboard"/>
  <IDevice id="mouse" type="GenericMouse"/>
  <ODevice id="screen" type="PCScreen"/>
  <ObjectHolder id="theCurrentViewpoint"/>
  <Object id="viewpoint" filename="" type="Viewpoint"/>
  <Binding iE="_self" iP="viewpoint" oE="theCurrentViewpoint" oP="object"/>

  <!-- Object to be modeled -->
  <Object id="clay" type="Material"/>

  <!-- Load Objects -->

  <!-- Move Viewpoint -->
  <!--   Rotation -->
  <Filter id="mouse2Ray" type="Mouse2Ray"/>
  <Filter id="getPoint" type="IntersectByBSphere"/>
  <Filter id="rotate" type="RotateByDragging"/>
  <Filter id="switchRot" type="Switch"/>
  <Binding iE="mouse" iP="mousePos" oE="mouse2Ray" oP="pos2D"/>
  <Binding iE="theCurrentViewpoint" iP="objectChanged"
oE="mouse2Ray" oP="viewpoint"/>
  <Binding iE="_self" iP="screen" oE="mouse2Ray" oP="screen"/>
  <Binding iE="_self" iP="clay" oE="getPoint" oP="object"/>
  <Binding iE="mouse2Ray" iP="ray" oE="getPoint" oP="ray"/>
  <Binding iE="clay" iP="currentBSphere" oE="getPoint" oP="compute"/>
  <Binding iE="getPoint" iP="point" oE="rotate" oP="point"/>
  <Binding iE="_self" iP="clay" oE="rotate" oP="object"/>
  <Binding iE="mouse" iP="lButtonPressed" oE="switch" oP="signalOn"/>
  <Binding iE="mouse" iP="lButtonReleased" oE="switch" oP="signalOff"/>
  <Binding iE="switch" iP="onOff" oE="rotate" oP="on"/>
  <Binding iE="switch" iP="onOff" oE="rotate" oP="flushState"/>

  <!--   Translation -->
  <Filter id="transXY" type="TranslatePlane">
  <Filter id="transXZ" type="TranslatePlane">
  <Filter id="switchXY" type="Switch"/>
  <Filter id="switchXZ" type="Switch"/>
  <Constant id="xy" value="XY"/>
  <Constant id="xz" value="XZ"/>
  <Constant id="scale" type="float" value="0.001"/>

  <Binding iE="mouse" iP="mButtonPressed" oE="switchXY" oP="signalOn"/>
  <Binding iE="mouse" iP="mButtonReleased" oE="switchXY" oP="signalOff"/>
  <Binding iE="switchXY" iP="onOff" oE="transXY" oP="on"/>
  <Binding iE="mouse" iP="mButtonReleased" oE="transXY" oP="flushState"/>
  <Binding iE="_self" iP="xy" oE="transXY" oP="plane"/>
  <Binding iE="mouse" iP="mousePos" oE="transXY" oP="pos2D"/>
  <Binding iE="_self" iP="clay" oE="transXY" oP="object"/>
  <Binding iE="_self" iP="scale" oE="transXY" oP="scaleFactor"/>

  <Binding iE="mouse" iP="rButtonPressed" oE="switchXZ" oP="signalOn"/>
  <Binding iE="mouse" iP="rButtonReleased" oE="switchXZ" oP="signalOff"/>
  <Binding iE="switchXZ" iP="onOff" oE="transXZ" oP="on"/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="mouse" iP="rButtonReleased" oE="transXZ" oP="flushState"/>
  <Binding iE="_self" iP="xz" oE="transXZ" oP="plane"/>
  <Binding iE="mouse" iP="mousePos" oE="transXZ" oP="pos2D"/>
  <Binding iE="_self" iP="clay" oE="transXZ" oP="object"/>
  <Binding iE="_self" iP="scale" oE="transXZ" oP="scaleFactor"/>

  <!-- Spin Material -->
  <!-- Move Tool -->
  <!-- Contact Feedback -->
  <!-- Cut Material -->
  <!-- Undo  -->
</App>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix C

DTD for InTml Files

The following is a transcript of the InTml DTD file. ENTITY elements define macros that can be reused in the following elements. Elements define which other elements can contain and available attributes. Contained elements are described in other entities in the document, and indicate their cardinality (no decoration is 1, * indicates 0 or more, and + 1 or more). The special keyword EMPTY defines an elements with no contained elements. Attributes indicate their name, their type, and their default value, if any. For more details, consult [26].

```
<!-- Interaction Techniques Markup Language    -->
<!-- Version 1.5                               -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
  May 21 2002:
        Constants have also a type, in order to allow fast development
of loaders (i.e. InTmlLoader).
  May 9 2002:
        Change App definition from
App ( (Import)*, (Overrides)*, (ODevice | IDevice)*, Platform?,
(Object | Constant | ObjectHolder)*, Filter*,
Binding* )
to
App ( (Import, Overrides, ODevice, IDevice, Platform,
Object, Constant, ObjectHolder, Filter,
Binding)* )
in order to allow the definition of elements in any order. The
semantics changes for Platform, which now has to be checked later on
for repetitions.
  May 7 2002:
        Policy is changed to IPorts, since it can't vary among bindings.
  Jan 24 2002:
        Name changed to InTml
Policy is added to Binding. It's semantic is application dependant,
 and three default names are defined: ADD (add events), ANY (pick any
 event), and AVE (average of the events).

  Check:
        Packages names... in FilterClassSet? in FilterClass? Import?
notation a.b.c...
Filter/Object/DeviceClass vs Filter/Object/Device
  June 8 2001:
        target/port changed to origin/destination in TARGET_DEF
Binding is limited to FilterClass and App. Removed from
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  Filter and Object.
Input is eliminated from ObjectHolder. Use Binding instead.
Input is eliminated from App. There are no aliases anymore.
Port is eliminated from IPort and OPort.
Port is eliminated from Binding. Instead, a more complex syntax
is used in binding. Now, origin and destination has
the form a.b.c[i]
origin/destination in Binding are changed to iE/iP/iI/oE/oP/oI
Scene is eliminated from App. It is now defined as an Object
with type Scene
IT is eliminated from App. It is now defined as Filter
A definition of a paper is taken out of this specification.
  It has to be defined by a companion DTD
  May 29 2001:
        FilterClassSet changed to Package
        Documentation goes to the desc file at docDTD. Using dtd2html from
http://www.oac.uci.edu/indiv/ehood/perlSGML.html
  May 25 2001:
        Integrate ITs to Filters
        Arrays of I/O ports
An interface can "Implement" another
Add ObjectClass as a definition for types that can be modified
Add DeviceClass
Add Platform
  March 2001:
InitArgs and Args in ITClass has been removed.
I/OPort manage Refs, in order to manage an ITClass interface.
        Support for applications added
Change of the project's name to 3dml
Website is now http://www.cs.ualberta.ca/~pfiguero/3dml/spec/3dml.dtd
The Binding group has been removed
Rules about content are included as invariants (inside comments)
  -->

<!ENTITY % ID
           "id   ID           #REQUIRED">
<!ENTITY % ID2
           "id   NMTOKEN      #REQUIRED">
<!ENTITY % ID3
           "id   NMTOKEN      #IMPLIED">
<!ENTITY % ID_TYPE
   "%ID2;
    type NMTOKEN #IMPLIED">
<!ENTITY % ID_TYPEREQ
   "%ID2;
    type NMTOKEN #REQUIRED">
<!ENTITY % ID_TYPE_DEF
   "%ID_TYPE;
    defValue CDATA #IMPLIED">
<!ENTITY % ID_TYPEREQ_DEF
   "%ID_TYPEREQ;
    defValue CDATA #IMPLIED">
<!ENTITY % DESC_INITO
   "ShortDesc?, Description?">
<!ENTITY % DESC_INIT
   "%DESC_INITO;, Indexes?, PaperRef*">
<!ENTITY % BOOLENUM
           "( true | false )">
<!ENTITY % ARRAY_DEF
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
   'isArray     (true|false)        "false"
    typeArray   (static|dynamic)    "dynamic"
    maxArray    NMTOKEN             #IMPLIED'>

<!-- ................................................................. -->
<!ELEMENT Package  (Import*, (FilterClass | ObjectClass | DeviceClass )+ )>
<!ATTLIST Package %ID3;>

<!-- ................................................................. -->
<!ELEMENT Import  EMPTY >
<!ATTLIST Import  %ID2; >

<!-- ................................................................. -->
<!ELEMENT FilterClass ( %DESC_INIT; , Implements*,
  (Filter | Object | ObjectHolder | Constant |
  Binding | IPort | OPort)* ) >
<!ATTLIST FilterClass  %ID; >

<!-- ................................................................. -->
<!ELEMENT ObjectClass  (%DESC_INIT; , Implements*, ( IPort | OPort )* ) >
<!ATTLIST ObjectClass  %ID; >

<!-- ................................................................. -->
<!ELEMENT DeviceClass  (%DESC_INIT; , Implements*, ( IPort | OPort)* ) >
<!ATTLIST DeviceClass  %ID; >

<!-- ................................................................. -->
<!ELEMENT ShortDesc (#PCDATA) >

<!-- ................................................................. -->
<!ELEMENT Description (#PCDATA) >

<!-- ................................................................. -->
<!ELEMENT Indexes  ( Index* ) >

<!-- ................................................................. -->
<!ELEMENT PaperRef  EMPTY >
<!ATTLIST PaperRef %ID2;
   detail NMTOKEN #REQUIRED >

<!-- ................................................................. -->
<!ELEMENT Implements  EMPTY >
<!ATTLIST Implements classId NMTOKEN #REQUIRED >

<!-- ................................................................. -->
<!ELEMENT Filter ( %DESC_INIT0; ) >
<!ATTLIST Filter %ID_TYPEREQ; >

<!-- ................................................................. -->
<!ELEMENT Object ( %DESC_INIT0; ) >
<!ATTLIST Object %ID_TYPEREQ;
     filename  CDATA   #IMPLIED
  primitive ( Box | Cone | Cylinder | Ellipse )  "Box"  >

<!-- ................................................................. -->
<!ELEMENT ObjectHolder EMPTY >
<!ATTLIST ObjectHolder %ID2; >

<!-- ................................................................. -->
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<!ELEMENT Binding  EMPTY >
<!ATTLIST Binding   iE   NMTOKEN #REQUIRED
                    iP   NMTOKEN #REQUIRED
                    iI   NMTOKEN #IMPLIED
            oE   NMTOKEN #REQUIRED
                    oP   NMTOKEN #REQUIRED
                    oI   NMTOKEN #IMPLIED >

<!-- ........................................................... -->
<!ELEMENT Index  EMPTY >
<!ATTLIST Index   id   NMTOKEN       #REQUIRED
  value NMTOKEN      #REQUIRED >

<!-- ........................................................... -->
<!ELEMENT IPort ( %DESC_INITO; ) >
<!ATTLIST IPort  %ID_TYPEREQ_DEF;
                 %ARRAY_DEF;
 policy CDATA #IMPLIED >

<!-- ........................................................... -->
<!ELEMENT OPort ( %DESC_INITO; ) >
<!ATTLIST OPort    %ID_TYPEREQ;
                   %ARRAY_DEF; >

<!-- ........................................................... -->
<!ELEMENT App ( %DESC_INIT; , (Import | Overrides | ODevice | IDevice |
                Platform |
Object | Constant | ObjectHolder | Filter |
Binding)* ) >
<!ATTLIST App %ID2; >

<!-- ........................................................... -->
<!ELEMENT ODevice ( %DESC_INITO; ) >
<!ATTLIST ODevice %ID_TYPEREQ; >

<!-- ........................................................... -->
<!ELEMENT IDevice ( %DESC_INITO; ) >
<!ATTLIST IDevice %ID_TYPEREQ; >

<!-- ........................................................... -->
<!ELEMENT Constant EMPTY >
<!ATTLIST Constant %ID_TYPE;
   value CDATA #REQUIRED >

<!-- ........................................................... -->
<!ELEMENT PlatformClass  (%DESC_INIT; , (Import)*, (IDevice | ODevice)* ) >
<!ATTLIST PlatformClass  %ID; >

<!-- ........................................................... -->
<!ELEMENT Platform EMPTY >
<!ATTLIST Platform %ID_TYPEREQ; >

<!-- Implementation tags. Just when the language is implemented .... -->

<!-- ........................................................... -->
<!ELEMENT Overrides  EMPTY >
<!ATTLIST Overrides classId NMTOKEN #REQUIRED >
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix D

Library of Reusable Concepts in InTml

These is a copy of the files that define the InTml library. They define basic VR–related devices, basic definition for geometric objects, control techniques, navigation techniques, selection techniques, feedback techniques, and a subset of the interaction techniques described in [8].

Devices

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
        "intml.dtd">
<!-- Copyright info
  devices: Device definition classes.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
First version
  -->
<!-- Devices in a VR application    -->
<Package id="devices">
  <DeviceClass id="Button">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <ShortDesc>Basic Button</ShortDesc>
  <Description>
    It models the states on a button. Check the model and change the
    simple output of a signal for a boolean. After this change the name
  </Description>
  <Indexes>
    <Index id="first" value="intml.Devices.Input.Abstract"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <OPort id="clicked" type="boolean">
    <ShortDesc>true when it's been clicked (pressed/released)</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="GenericJoystick">
  <ShortDesc>Generic joystick definition</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input.Abstract"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <OPort id="xPos" type="int">
    <ShortDesc>Position left-right of the joystick</ShortDesc>
  </OPort>
  <OPort id="yPos" type="int">
    <ShortDesc>Position front-back of the joystick</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos2D">
    <ShortDesc>(xPos, yPos)</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="SideWinderPro">
  <ShortDesc>Microsoft&apos;s SideWinder joystick</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="GenericJoystick"/>
  <OPort id="buttonClicked" isArray="true"
         maxArray="8" type="boolean" typeArray="static">
    <ShortDesc>Available buttons in the SideWinderPro</ShortDesc>
  </OPort>
  <OPort id="hatSwitchClicked" isArray="true" maxArray="8"
         type="boolean" typeArray="static">
    <ShortDesc>Directional control</ShortDesc>
    <Description>  It provides directional control with the thumb. The
      output buttons are  arranged counter-clockwise, starting from
      front (left-front, left, ...)     </Description>
  </OPort>
  <OPort id="throttle" type="int">
    <ShortDesc>A value that indicates an incremental action.</ShortDesc>
  </OPort>
  <OPort id="rotation" type="int">
    <ShortDesc>A value that indicates the rotation of the joystick.</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="Generic3DOFTracker">
  <ShortDesc>A generic orientation tracker</ShortDesc>
  <Description/>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Indexes>
    <Index id="first" value="intml.Devices.Input.Abstract"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <OPort id="q" type="Quaternion">
    <ShortDesc>General orientation of the tracker</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="Generic6DOFTracker">
  <ShortDesc>A generic position and orientation tracker</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input.Abstract"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="Generic3DOFTracker"/>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>3D position of the tracker</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="Insidetrack">
  <ShortDesc>Polhemus insidetrack 6DOF tracker</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="Generic6DOFTracker"/>
</DeviceClass>
<DeviceClass id="InterSenseWandTracker">
  <ShortDesc>Hand tracker from InterSense</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="GenericJoystick"/>
  <Implements classId="Generic6DOFTracker"/>
  <OPort id="buttonClicked" isArray="true" maxArray="4"
         type="boolean" typeArray="static">
    <ShortDesc>Available buttons in the wand.</ShortDesc>
  </OPort>
</DeviceClass>
<DeviceClass id="InterSenseHeadTracker">
  <ShortDesc>Head tracker from InterSense</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="Generic6DOFTracker"/>
</DeviceClass>
<DeviceClass id="GenericScreen">
  <ShortDesc>A generic output screen</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
</DeviceClass>
<DeviceClass id="FishTankScreen">
  <ShortDesc>A 3D output screen with viewpoint position</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="I-glasses">
  <ShortDesc>Virtual i-glasses HMD</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="VisroomScreens">
  <ShortDesc>Screens at the Visroom</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="VisroomScreen">
  <ShortDesc>A Particular screen at the Visroom</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="GenericHMD">
  <ShortDesc>A generic HMD</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output.Abstract"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="V6HMD">
  <ShortDesc>Virtual Research - V6 HMD</ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Output"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
</DeviceClass>
<DeviceClass id="GenericMouse">
  <ShortDesc>A generic mouse definition </ShortDesc>
  <Description/>
  <Indexes>
    <Index id="first" value="intml.Devices.Input"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <OPort id="xPos" type="int">
    <ShortDesc>Position in X.</ShortDesc>
  </OPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <OPort id="yPos" type="int">
      <ShortDesc>Position in Y.</ShortDesc>
    </OPort>
    <OPort id="mouseMove" type="Pos2D">
      <ShortDesc>Absolute position of the mouse pointer</ShortDesc>
      <Description>Adds up xPos and yPos into one output.</Description>
    </OPort>
    <OPort id="rButtonDn" type="boolean">
      <ShortDesc>Right button pressed.</ShortDesc>
    </OPort>
    <OPort id="mButtonDn" type="boolean">
      <ShortDesc>Medium button pressed.</ShortDesc>
    </OPort>
    <OPort id="lButtonDn" type="boolean">
      <ShortDesc>Left button pressed.</ShortDesc>
    </OPort>
    <OPort id="rButtonUp" type="boolean">
      <ShortDesc>Right button released.</ShortDesc>
    </OPort>
    <OPort id="mButtonUp" type="boolean">
      <ShortDesc>Medium button released.</ShortDesc>
    </OPort>
    <OPort id="lButtonUp" type="boolean">
      <ShortDesc>Left button released.</ShortDesc>
    </OPort>
  </DeviceClass>
  <DeviceClass id="WheelMouse">
    <ShortDesc>A mouse with a wheel for scrolling</ShortDesc>
    <Description/>
    <Indexes>
      <Index id="first" value="intml.Devices.Input"/>
      <Index id="papers" value="_hidden"/>
    </Indexes>
    <Implements classId="GenericMouse"/>
  </DeviceClass>
  <DeviceClass id="GenericKeyboard">
    <ShortDesc>A generic keyboard definition</ShortDesc>
    <Description/>
    <Indexes>
      <Index id="first" value="intml.Devices.Input"/>
      <Index id="papers" value="_hidden"/>
    </Indexes>
    <OPort id="asciiKeyClicked" isArray="true" maxArray="256"
          type="boolean" typeArray="static">
      <ShortDesc>A button for each ASCII code in the keyboard.</ShortDesc>
    </OPort>
  </DeviceClass>
  <DeviceClass id="MicroScribe3DFX">
    <ShortDesc>Immersion - MicroScribe 3DFX 3d digitizer </ShortDesc>
    <Description/>
    <Indexes>
      <Index id="first" value="intml.Devices.Input"/>
      <Index id="papers" value="_hidden"/>
    </Indexes>
    <OPort id="pos" type="Pos3D">
      <ShortDesc>The current position.</ShortDesc>
    </OPort>
  </DeviceClass>
</Package>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Object Definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
    "intml.dtd">
<!-- Copyright info
  objects: Generic object definitions.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
First version
  -->

<!-- Main types in a VR application    -->

<Package id="objects">

<ObjectClass id="VRObject">
  <ShortDesc>Interface for a VR object in the dataflow</ShortDesc>
  <Description>
VRObjectClass defines the input and output ports of any VRObject
in the dataflow.
        Transformations are organized as follows:
          Rotations, scales, translations
  </Description>
  <Indexes>
    <Index id="first" value="intml.Object"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <IPort id="setPos" type="Pos3D">
    <ShortDesc>Changes the world position of an object</ShortDesc>
    <Description>
Changes the world-related position of an object. It is considered a relative
position if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="posChanged" type="Pos3D">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <ShortDesc>Informs when the object moves</ShortDesc>
  </OPort>
  <IPort id="setQ" type="Quaternion">
    <ShortDesc>Changes the world orientation of an object</ShortDesc>
    <Description>
Changes the world-related orientation of an object. It is considered a relative
orientation if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="qChanged" type="Quaternion">
    <ShortDesc>Informs when the object rotates</ShortDesc>
  </OPort>
  <IPort id="setScale" type="Vector3">
    <ShortDesc>Changes the scale of an object</ShortDesc>
    <Description>
Changes the size of an object. It is considered a relative
size if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="scaleChanged" type="Vector3">
    <ShortDesc>Informs when the object changes its size</ShortDesc>
  </OPort>
  <IPort id="setMatrix" type="Matrix4">
    <ShortDesc>Changes the rigid transformations.</ShortDesc>
    <Description>
        Changes the rigid transformations that apply to the object.
It is useful when it necessary to apply rigid transformations
at once and at a specific order.
    </Description>
  </IPort>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds a new part to this object</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="VRObject">
    <ShortDesc>Informs when a part is added to the object</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Removes a part from this object</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="VRObject">
    <ShortDesc>Informs when a part is removed from the object</ShortDesc>
  </OPort>
  <IPort id="setBB" type="boolean">
    <ShortDesc>Defines if the bounding box is visible or not</ShortDesc>
  </IPort>
  <OPort id="BBChanged" type="boolean">
    <ShortDesc>Informs changes in the bounding box visibility</ShortDesc>
  </OPort>
  <IPort id="setColor" type="Color">
    <ShortDesc>Changes the main color of an object</ShortDesc>
  </IPort>
  <OPort id="colorChanged" type="Color">
    <ShortDesc>Informs when the color changes</ShortDesc>
  </OPort>
</ObjectClass>

<ObjectClass id="Scene">
  <ShortDesc>Interface for a Scene in the dataflow</ShortDesc>
  <Description>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      A Scene is a set of unstructured geometry that can't be selected, and a
      set of VRobjects, which can be selected. Its implementation defines the
      general structures required for selection computation.
  </Description>
  <Indexes>
    <Index id="first" value="intml.Object"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Implements classId="VRObject"/>
</ObjectClass>


<ObjectClass id="VRSystem">
  <ShortDesc>VR runtime engine</ShortDesc>
  <Description>
      The VRSystem represents the virtual machine that runs a particular
      application. It's interface represents all possible modifications
      in a virtual environment (VE).
  </Description>
  <Indexes>
    <Index id="first" value="intml.Object"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds an object to the VE</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="VRObject">
    <ShortDesc>Informs when a part is added to the VE</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Deletes an object to the VE</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="VRObject">
    <ShortDesc>Informs when an object is deleted from the VE</ShortDesc>
  </OPort>
  <IPort id="setScene" type="Scene">
    <ShortDesc>Defines the current scene.</ShortDesc>
    <Description>
      Defines the current scene. There is only one active scene in a VE.
    </Description>
  </IPort>
  <OPort id="sceneChanged" type="Scene">
    <ShortDesc>Informs when the scene is changed.</ShortDesc>
  </OPort>
  <IPort id="addFilter" type="FilterClass">
    <ShortDesc>Adds a new unconnected filter to the VE</ShortDesc>
  </IPort>
  <OPort id="filterAdded" type="FilterClass">
    <ShortDesc>Informs when a filter is added to the VE</ShortDesc>
  </OPort>
  <IPort id="removeFilter" type="FilterClass">
    <ShortDesc>Removes an unconnected filter from the VE</ShortDesc>
  </IPort>
  <OPort id="filterRemoved" type="FilterClass">
    <ShortDesc>Informs when a filter is removed from the VE</ShortDesc>
  </OPort>
  <IPort id="addConnection" type="Binding">
    <ShortDesc>Adds a new connection between filters in the VE</ShortDesc>
  </IPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <OPort id="connectionAdded" type="Binding">
    <ShortDesc>Informs when a connection is added to the VE</ShortDesc>
  </OPort>
  <IPort id="removeConnection" type="Binding">
    <ShortDesc>Removes a connection between filters in the VE</ShortDesc>
  </IPort>
  <OPort id="connectionRemoved" type="Binding">
    <ShortDesc>Informs when a connection is deleted from the VE</ShortDesc>
  </OPort>
</ObjectClass>

</Package>
```

Application Control Techniques

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
    "intml.dtd">
<!-- Copyright info
  control: InTs for application control

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
First version
  -->

<!-- General Control Techniques    -->

<Package id="ualberta.control">

<FilterClass id="QuitbyButton">
<ShortDesc>Quit when a button is pressed</ShortDesc>
  <Description>
It terminates the application
when the button is pressed.
 </Description>
  <Indexes>
    <Index id="first"
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
value="intml.control"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="bClicked" type="boolean">
    <ShortDesc>Button that quits the application</ShortDesc>
  </IPort>
</FilterClass>

</Package>
```

Navigation Techniques

```
<?xml version="1.0"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
    "intml.dtd">
<!-- Copyright info
  travel: Travel interaction techniques.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
First version
  -->

<!-- Travel Techniques    -->

<Package id="travel">
<Import id="objects"/>

<FilterClass id="SteerHead">
  <IPort id="headPos" type="Pos3D" defValue="0, 0, 0"></IPort>
  <IPort id="headQ" type="Quaternion" defValue="0 ,0 , 0, 1"></IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0"></IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1"></IPort>
  <OPort id="pos" type="Pos3D"></OPort>
  <OPort id="q" type="Quaternion"></OPort>
</FilterClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<FilterClass id="MoveUnrestricted">
 <ShortDesc>Move at a certain speed from an starting position</ShortDesc>
  <Description>
MoveUnrestricted computes a
new position, given a starting
point, a direction, and a certain speed.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.travel"/>
    <Index id="papers"
value="_unknown"/>
  </Indexes>
  <IPort id="direction" type="Quaternion" defValue="0 ,0 , 0, 1">
    <ShortDesc>Direction of movement</ShortDesc>
  </IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
     <ShortDesc>Starting position</ShortDesc>
  </IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1">
    <ShortDesc>Starting orientation</ShortDesc>
  </IPort>
  <IPort id="speed" type="float" defValue="1.0">
    <ShortDesc>Speed of movement</ShortDesc>
  </IPort>
  <IPort id="timer" type="int">
    <ShortDesc>Number of seconds</ShortDesc>
    <Description>
The first value received is
taken as time 0.
    </Description>
  </IPort>
  <OPort id="pos" type="Pos3D">
    <ShortDesc>Computed position</ShortDesc>
  </OPort>
  <OPort id="q" type="Quaternion">
    <ShortDesc>Computed orientation</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="MoveInGround">
  <IPort id="direction" type="Quaternion" defValue="0 ,0 , 0, 1"></IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0"></IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1"></IPort>
  <IPort id="speed" type="float" defValue="1.0"></IPort>
  <IPort id="plane" type="Plane3D" defValue="0, 0, 0, 1, 0, 0"></IPort>
  <OPort id="pos" type="Pos3D"></OPort>
  <OPort id="q" type="Quaternion"></OPort>
</FilterClass>

<FilterClass id="Walk">
  <IPort id="p" type="Pos3D" defValue="0, 0, 0"></IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0"></IPort>
  <OPort id="pos" type="Pos3D"></OPort>
</FilterClass>

<FilterClass id="SteeringIT">
  <Filter id="steer" type="SteerHead"></Filter>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <ObjectHolder id="v"/>

  <IPort id="pos" type="Pos3D">
  </IPort>
  <IPort id="q" type="Quaternion">
  </IPort>
  <IPort id="viewpoint" type="VRObject">
  </IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
  </IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1">
  </IPort>

  <Binding iE="steer" iP="pos" oE="v" oP="setPos" />
  <Binding iE="steer" iP="q" oE="v" oP="setQ" />
  <Binding iE="_self" iP="pos" oE="steer" oP="headPos" />
  <Binding iE="_self" iP="q" oE="steer" oP="headQ" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="_self" iP="posOffset" oE="steer" oP="posOffset" />
  <Binding iE="_self" iP="qOffset" oE="steer" oP="qOffset" />
</FilterClass>

<FilterClass id="FlyUnrestrictedIT">
  <Filter id="fly" type="MoveUnrestricted"></Filter>
  <ObjectHolder id="v"/>

  <IPort id="q" type="Quaternion">
  </IPort>
  <IPort id="viewpoint" type="VRObject">
  </IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
  </IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1">
  </IPort>
  <IPort id="speed" type="float" defValue="1.0">
  </IPort>

  <Binding iE="fly" iP="pos" oE="v" oP="setPos" />
  <Binding iE="fly" iP="q" oE="v" oP="setQ" />
  <Binding iE="_self" iP="q" oE="fly" oP="direction" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="_self" iP="posOffset" oE="fly" oP="posOffset" />
  <Binding iE="_self" iP="qOffset" oE="fly" oP="qOffset" />
  <Binding iE="_self" iP="speed" oE="fly" oP="speed" />
</FilterClass>

<FilterClass id="FlyInPlaneIT">
  <Filter id="fly" type="MoveInGround"></Filter>
  <ObjectHolder id="v"/>

  <IPort id="q" type="Quaternion">
  </IPort>
  <IPort id="viewpoint" type="VRObject">
  </IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
  </IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0 ,0, 0, 1">
  </IPort>
  <IPort id="speed" type="float" defValue="1.0">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </IPort>
  <IPort id="plane" type="Plane3D" defValue="0, 0, 0, 1, 0, 0">
  </IPort>

  <Binding iE="fly" iP="pos" oE="v" oP="setPos" />
  <Binding iE="fly" iP="q" oE="v" oP="setQ" />
  <Binding iE="_self" iP="q" oE="fly" oP="direction" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="_self" iP="posOffset" oE="fly" oP="posOffset" />
  <Binding iE="_self" iP="qOffset" oE="fly" oP="qOffset" />
  <Binding iE="_self" iP="speed" oE="fly" oP="speed" />
  <Binding iE="_self" iP="plane" oE="fly" oP="plane" />
</FilterClass>

<FilterClass id="WalkIT">
  <Filter id="walk" type="Walk"></Filter>
  <ObjectHolder id="v" />

  <IPort id="p" type="Pos3D">
  </IPort>
  <IPort id="viewpoint" type="VRObject">
  </IPort>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
  </IPort>

  <Binding iE="walk" iP="pos" oE="v" oP="setPos" />
  <Binding iE="_self" iP="p" oE="walk" oP="p" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="_self" iP="posOffset" oE="walk" oP="posOffset" />
</FilterClass>

</Package>
```

Selection and Feedback Techniques

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
    "intml.dtd">
<!-- Copyright info
  selection: Selection interaction techniques.

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
  June 10 2001:
        Changed to the new specification
  Mayo 2001:
First version
  -->

<!-- Selection Techniques    -->

<Package id="ualberta.selection">
<Import id="objects"/>

<!-- This class has to be defined in the implementation -->
<FilterClass id="SelectByTouching">
  <Description>
    Implements details of selection by collision detection.
    It receives position and orientation updates of handRepr
    and it computes which object from the scene is intersected
    by handRepr. It allows dynamic changes to the scene by
    connecting the ports addObject and removeObject.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.selection.details"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="p" type="Pos3D" defValue="handRepr.posChanged">
<ShortDesc>Change of position</ShortDesc>
  </IPort>
  <IPort id="q" type="Quaternion" defValue="handRepr.qChanged">
<ShortDesc>Change of rotation</ShortDesc>
  </IPort>
  <IPort id="handRepr" type="VRObject">
<ShortDesc>Object that represents the users' hand</ShortDesc>
  </IPort>
  <IPort id="scene" type="Scene">
<ShortDesc>Selectable objects</ShortDesc>
  </IPort>
  <IPort id="addObjectToScene" type="VRObject">
<ShortDesc>Dynamically added objects</ShortDesc>
  </IPort>
  <IPort id="removeObjectFromScene" type="VRObject">
<ShortDesc>Dynamically removed objects</ShortDesc>
  </IPort>
  <OPort id="object" type="VRObject">
<ShortDesc>Selected object</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="SelectByRay">
  <Description>
    Implements details of selection by intersection with a ray.
    It receives position and orientation updates of a ray in the scene
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    and it computes which object from the scene is intersected first.
    It allows dynamic changes to the scene by
    connecting the ports addObject and removeObject.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.selection.details"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="p" type="Pos3D" defValue="0, 0, 0">
<ShortDesc>Position of the ray</ShortDesc>
  </IPort>
  <IPort id="q" type="Quaternion" defValue="0, 0, 0, 1">
<ShortDesc>Rotation of the ray</ShortDesc>
  </IPort>
  <IPort id="scene" type="Scene">
<ShortDesc>Selectable objects</ShortDesc>
  </IPort>
  <IPort id="addObjectToScene" type="VRObject">
<ShortDesc>Dynamically added objects</ShortDesc>
  </IPort>
  <IPort id="removeObjectFromScene" type="VRObject">
<ShortDesc>Dynamically removed objects</ShortDesc>
  </IPort>
  <OPort id="object" type="VRObject">
<ShortDesc>Selected object</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="FeedbackOne">
  <ShortDesc>Changes the appearance of an object</ShortDesc>
  <Description>
    It changes the appearance of the object that
    is received in its input port. It is useful
    as feedback of an operation that selects one
    object. It is meant to have two object holders as output,
    one for the current object and the other for the previous
    object, in order to change back its state.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.feedback.details"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="obj" type="VRObject">
<ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="type" type="String" defValue="boundingBox">
<ShortDesc>Type of change.</ShortDesc>
        <Description>
  Type of change. It can be "boundingBox" or "color"
</Description>
  </IPort>
  <IPort id="color" type="Color" defValue="1, 1, 1">
<ShortDesc>Feedback color</ShortDesc>
        <Description>
  Feedback color. If type is "color" the object will be
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  changed to this color.
</Description>
  </IPort>
  <OPort id="currentObject" type="VRObject">
<ShortDesc>Object to be changed.</ShortDesc>
  </OPort>
  <OPort id="setBBCurrent" type="boolean">
<ShortDesc>Current value for the BB.</ShortDesc>
  </OPort>
  <OPort id="setColorCurrent" type="Color">
<ShortDesc>Current value for color.</ShortDesc>
  </OPort>
  <OPort id="previousObject" type="VRObject">
<ShortDesc>Previous changed object.</ShortDesc>
  </OPort>
  <OPort id="setBBPrevious" type="boolean">
<ShortDesc>Previous value for the BB.</ShortDesc>
  </OPort>
  <OPort id="setColorPrevious" type="Color">
<ShortDesc>Previous value for color.</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="FeedbackSet">
  <ShortDesc>Changes the appearance of a set of objects</ShortDesc>
  <Description>
    It changes the appearance of the set of objects that
    is received in its input port. It is useful
    as feedback of an operation that selects a set of
    objects. It is meant to have two object holders as output,
    one for the current set and the other for the previous,
    in order to change back its state.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.feedback.details"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="obj" type="Scene">
<ShortDesc>Set of objects to be changed</ShortDesc>
  </IPort>
  <IPort id="type" type="String" defValue="boundingBox">
<ShortDesc>Type of change.</ShortDesc>
        <Description>
  Type of change. It can be "boundingBox" or "color"
</Description>
  </IPort>
  <IPort id="color" type="Color" defValue="1, 1, 1">
<ShortDesc>Feedback color</ShortDesc>
        <Description>
  Feedback color. If type is "color" the objects will be
  changed to this color.
</Description>
  </IPort>
  <OPort id="currentObject" type="Scene">
<ShortDesc>Objects to be changed.</ShortDesc>
  </OPort>
  <OPort id="setBBCurrent" type="boolean">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<ShortDesc>Current value for the BB.</ShortDesc>
  </OPort>
  <OPort id="setColorCurrent" type="Color">
<ShortDesc>Current value for color.</ShortDesc>
  </OPort>
  <OPort id="previousObject" type="Scene">
<ShortDesc>Previous changed objects.</ShortDesc>
  </OPort>
  <OPort id="setBBPrevious" type="boolean">
<ShortDesc>Previous value for the BB.</ShortDesc>
  </OPort>
  <OPort id="setColorPrevious" type="Color">
<ShortDesc>Previous value for color.</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="FeedbackScaleSet">
  <ShortDesc>Changes the position and scale of a set of objects</ShortDesc>
  <Description>
    It changes the scale and position of a set of objects,
    according to the rules of the IT "ScaleGrab". It is meant to be used
    with two holders as output, one for the current selected set of
    objects, and the other for the previous one.
  </Description>
  <Indexes>
    <Index id="first" value="intml.feedback.details"/>
    <Index id="papers" value="mine97"/>
  </Indexes>
  <IPort id="pHead" type="Pos3D" defValue="0, 1, 0">
<ShortDesc>Head position.</ShortDesc>
  </IPort>
  <IPort id="qHead" type="Quaternion" defValue="0, 0, 0, 1">
<ShortDesc>Head orientation.</ShortDesc>
  </IPort>
  <IPort id="obj" type="Scene">
<ShortDesc>Selected objects.</ShortDesc>
  </IPort>
  <IPort id="maxRadius" type="float" defValue="1.0">
<ShortDesc>Maximum radius of the scaled objects.</ShortDesc>
  </IPort>
  <OPort id="currentObjects" type="Scene">
<ShortDesc>Current selected objects.</ShortDesc>
  </OPort>
  <OPort id="setMatrixCurrent" type="Matrix4">
<ShortDesc>Current scale for the set.</ShortDesc>
  </OPort>
  <OPort id="previousObjects" type="Scene">
<ShortDesc>Previous set of selected objects</ShortDesc>
  </OPort>
  <OPort id="setMatrixPrevious" type="Matrix4">
<ShortDesc>Previous scale.</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="GoGo">
  <ShortDesc>Lengthening behavior of GoGoIT.</ShortDesc>
  <Description>
      GoGo computes a new hand position as a function of the distance
      of the real hand from the chest, K, and D.
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </Description>
  <Indexes>
    <Index id="first" value="intml.selection.details"/>
    <Index id="papers" value="poupyrev96"/>
  </Indexes>
  <IPort id="pHead" type="Pos3D" defValue="0, 1, 0">
<ShortDesc>Head position.</ShortDesc>
  </IPort>
  <IPort id="pHand" type="Pos3D" defValue="0, 0, -1">
<ShortDesc>Hand position.</ShortDesc>
  </IPort>
  <IPort id="K" type="float" defValue="0.5">
<ShortDesc>Coefficient k (0..1)</ShortDesc>
  </IPort>
  <IPort id="D" type="float" defValue="0.6">
<ShortDesc>Distance for lenghtening</ShortDesc>
    <Description>
      Minimum distance for lengthening behavior. Usually 2/3 of the
      user's arm length.
    </Description>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>New hand position.</ShortDesc>
  </OPort>
  <OPort id="setVisible" type="boolean">
<ShortDesc>true if lengthening behavior is inactive.</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="Spotlight">
  <ShortDesc>Main computation for selection by a spotlight.</ShortDesc>
  <Description>
      Spotlight computes the direction and angle of the spotlight
      cone that starts in the user's head. It returns a set of objects
      that are inside the cone.
  </Description>
  <Indexes>
    <Index id="first" value="intml.selection.details"/>
    <Index id="papers" value="liang93"/>
  </Indexes>
  <IPort id="pHead" type="Pos3D" defValue="0, 0, 0">
    <ShortDesc>Head position</ShortDesc>
  </IPort>
  <IPort id="qHead" type="Quaternion" defValue="0, 0, 0, 1">
    <ShortDesc>Head orientation</ShortDesc>
  </IPort>
  <IPort id="pHand" type="Pos3D" defValue="0, -1, 0">
    <ShortDesc>Hand position</ShortDesc>
  </IPort>
  <IPort id="qHand" type="Quaternion" defValue="0, 0, 0, 1">
    <ShortDesc>Hand orientation</ShortDesc>
  </IPort>
  <IPort id="a" type="float" defValue="1.0">
    <ShortDesc></ShortDesc>
  </IPort>
  <IPort id="scene" type="Scene" defValue="">
    <ShortDesc>Objects to be selected</ShortDesc>
  </IPort>
  <IPort id="addObjectToScene" type="VRObject">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<ShortDesc>Dynamically added objects</ShortDesc>
  </IPort>
  <IPort id="removeObjectFromScene" type="VRObject">
<ShortDesc>Dynamically removed objects</ShortDesc>
  </IPort>
  <OPort id="object" type="Scene">
    <ShortDesc>Selected objects</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="FeedbackOneIT">
  <ShortDesc>Changes the appearance of a selected object</ShortDesc>
  <Description>
    It changes the appearance of an object. It has memory, so the previously
    selected object restores its previous appearance once a new object is selected.
  </Description>
  <Indexes>
    <Index id="first" value="intml.feedback"/>
    <Index id="papers" value="_hidden"/>
  </Indexes>
  <Filter id="feedback" type="FeedbackOne">
    <ShortDesc>Main computation</ShortDesc>
  </Filter>
  <ObjectHolder id="current" />
  <ObjectHolder id="previous" />

  <IPort id="obj" type="VRObject">
    <ShortDesc>Object to be changed</ShortDesc>
  </IPort>
  <IPort id="type" type="String">
    <ShortDesc>Type of change</ShortDesc>
  </IPort>
  <IPort id="color" type="Color">
    <ShortDesc>Color for the changed object</ShortDesc>
  </IPort>

  <Binding iE="feedback" iP="setBBCurrent" oE="current" oP="setBB" />
  <Binding iE="feedback" iP="setColorCurrent" oE="current" oP="setColor" />
  <Binding iE="feedback" iP="setBBPrevious" oE="previous" oP="setBB" />
  <Binding iE="feedback" iP="setColorPrevious" oE="previous" oP="setColor" />
  <Binding iE="feedback" iP="currentObject" oE="current" oP="object" />
  <Binding iE="feedback" iP="previousObject" oE="previous" oP="object" />
  <Binding iE="_self" iP="obj" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="type" oE="feedback" oP="type" />
  <Binding iE="_self" iP="color" oE="feedback" oP="color" />
</FilterClass>

<FilterClass id="FeedbackSetIT">
  <Filter id="feedback" type="FeedbackSet"></Filter>
  <ObjectHolder id="current" />
  <ObjectHolder id="previous" />

  <IPort id="obj" type="Scene"></IPort>
  <IPort id="type" type="String"></IPort>
  <IPort id="color" type="Color"></IPort>

  <Binding iE="feedback" iP="setBBCurrent" oE="current" oP="setBB" />
  <Binding iE="feedback" iP="setColorCurrent" oE="current" oP="setColor" />
  <Binding iE="feedback" iP="setBBPrevious" oE="previous" oP="setBB" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="feedback" iP="setColorPrevious" oE="previous" oP="setColor" />
  <Binding iE="feedback" iP="currentObject" oE="current" oP="object" />
  <Binding iE="feedback" iP="previousObject" oE="previous" oP="object" />
  <Binding iE="_self" iP="obj" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="type" oE="feedback" oP="type" />
  <Binding iE="_self" iP="color" oE="feedback" oP="color" />
</FilterClass>

<FilterClass id="FeedbackScaleSetIT">
  <Filter id="feedback" type="FeedbackScaleSet"></Filter>
  <ObjectHolder id="current" />
  <ObjectHolder id="previous" />

  <IPort id="obj" type="Scene"></IPort>
  <IPort id="maxRadius" type="float"></IPort>
  <IPort id="pHead" type="Pos3D"></IPort>
  <IPort id="qHead" type="Quaternion"></IPort>

  <Binding iE="feedback" iP="setMatrixCurrent" oE="current" oP="setMatrix" />
  <Binding iE="feedback" iP="setMatrixPrevious" oE="previous" oP="setMatrix" />
  <Binding iE="feedback" iP="currentObjects" oE="current" oP="object" />
  <Binding iE="feedback" iP="previousObjects" oE="previous" oP="object" />
  <Binding iE="_self" iP="obj" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="maxRadius" oE="feedback" oP="maxRadius" />
  <Binding iE="_self" iP="pHead" oE="feedback" oP="pHead" />
  <Binding iE="_self" iP="qHead" oE="feedback" oP="qHead" />
</FilterClass>

<FilterClass id="SelectByTouchingIT">
  <Filter id="select" type="SelectByTouching"></Filter>
  <Filter id="feedback" type="FeedbackOneIT"></Filter>
  <ObjectHolder id="hand" />

  <IPort id="pos" type="Pos3D"></IPort>
  <IPort id="q" type="Quaternion"></IPort>
  <IPort id="handRepr" type="VRObject"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <IPort id="addObjectToScene" type="VRObject"></IPort>
  <IPort id="removeObjectFromScene" type="VRObject"></IPort>
  <IPort id="type" type="String"></IPort>
  <IPort id="color" type="Color"></IPort>
  <OPort id="object" type="VRObject"></OPort>

  <Binding iE="select" iP="object" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="pos" oE="select" oP="p" />
  <Binding iE="_self" iP="pos" oE="hand" oP="p" />
  <Binding iE="_self" iP="q" oE="select" oP="q" />
  <Binding iE="_self" iP="q" oE="select" oP="q" />
  <Binding iE="_self" iP="handRepr" oE="select" oP="handRepr" />
  <Binding iE="_self" iP="handRepr" oE="hand" oP="object" />
  <Binding iE="_self" iP="scene" oE="select" oP="scene" />
  <Binding iE="_self" iP="addObjectToScene" oE="select" oP="addObjectToScene" />
  <Binding iE="_self" iP="removeObjectFromScene" oE="select" oP="removeObjectFromScene" />
  <Binding iE="_self" iP="type" oE="feedback" oP="type" />
  <Binding iE="_self" iP="color" oE="feedback" oP="color" />
  <Binding iE="select" iP="object" oE="_self" oP="object" />
</FilterClass>

<FilterClass id="SelectByRayIT">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<ShortDesc>Selection in a particular direction</ShortDesc>
  <Description>
SelectByRayIT selects an object
 from a set given a position and
a direction.
Pending: Add a ray as feedback
  </Description>
  <Indexes>
    <Index id="first"
value="intml.selection"/>
    <Index id="papers"
value="_unknown"/>
  </Indexes>
  <Filter id="select" type="SelectByRay"></Filter>
  <Filter id="feedback" type="FeedbackOneIT"></Filter>

  <IPort id="pos" type="Pos3D"></IPort>
  <IPort id="q" type="Quaternion"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <IPort id="addObjectToScene" type="VRObject"></IPort>
  <IPort id="removeObjectToScene" type="VRObject"></IPort>
  <IPort id="type" type="String"></IPort>
  <IPort id="color" type="Color"></IPort>
  <OPort id="object" type="VRObject"></OPort>

  <Binding iE="select" iP="object" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="pos" oE="select" oP="p" />
  <Binding iE="_self" iP="q" oE="select" oP="q" />
  <Binding iE="_self" iP="scene" oE="select" oP="scene" />
  <Binding iE="_self" iP="addObjectToScene" oE="select" oP="addObjectToScene" />
  <Binding iE="_self" iP="removeObjectToScene" oE="select" oP="removeObjectFromScene" />
  <Binding iE="_self" iP="type" oE="feedback" oP="type" />
  <Binding iE="_self" iP="color" oE="feedback" oP="color" />
  <Binding iE="select" iP="object" oE="_self" oP="object" />
</FilterClass>

<FilterClass id="GoGoIT">
  <Filter id="gogo" type="GoGo"></Filter>
  <Filter id="select" type="SelectByTouchingIT"></Filter>
  <Object id="cubeObj" type="VRObject" primitive="Box"></Object>
  <ObjectHolder id="cube" />

  <IPort id="K" type="float" defValue="0.167"></IPort>
  <IPort id="D" type="float" defValue="0.6"></IPort>
  <IPort id="posHead" type="Pos3D"></IPort>
  <IPort id="qHead" type="Quaternion"></IPort>
  <IPort id="posHand" type="Pos3D"></IPort>
  <IPort id="qHand" type="Quaternion"></IPort>
  <IPort id="handRepr" type="VRObject"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <IPort id="addObjectToScene" type="VRObject"></IPort>
  <IPort id="removeObjectFromScene" type="VRObject"></IPort>
  <IPort id="type" type="String"></IPort>
  <IPort id="color" type="Color"></IPort>
  <OPort id="object" type="VRObject"></OPort>

  <Binding iE="gogo" iP="setVisible" oE="cube" oP="setVisible" />
  <Binding iE="_self" iP="cubeObj" oE="cube" oP="object" />
  <Binding iE="_self" iP="cubeObj" oE="cube" oP="object" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="gogo" iP="pos" oE="select" oP="pos" />
  <Binding iE="_self" iP="qHand" oE="select" oP="q" />
  <Binding iE="_self" iP="K" oE="gogo" oP="K" />
  <Binding iE="_self" iP="D" oE="gogo" oP="D" />
  <Binding iE="_self" iP="posHead" oE="gogo" oP="pHead" />
  <Binding iE="_self" iP="posHand" oE="gogo" oP="pHand" />
  <Binding iE="_self" iP="posHand" oE="cube" oP="p" />
  <Binding iE="_self" iP="qHand" oE="select" oP="q" />
  <Binding iE="_self" iP="qHand" oE="cube" oP="q" />
  <Binding iE="_self" iP="handRepr" oE="select" oP="handRepr" />
  <Binding iE="_self" iP="scene" oE="select" oP="scene" />
  <Binding iE="_self" iP="addObjectToScene" oE="select" oP="addObjectToScene" />
  <Binding iE="_self" iP="removeObjectFromScene" oE="select" oP="removeObjectFromScene" />
  <Binding iE="_self" iP="type" oE="select" oP="type" />
  <Binding iE="_self" iP="color" oE="select" oP="color" />
  <Binding iE="select" iP="object" oE="_self" oP="object" />
</FilterClass>

<FilterClass id="SpotlightIT">
<ShortDesc>Select objects with a spotlight</ShortDesc>
  <Description>
It selects a set of objects, given
a direction and a radius of a cone
centered in such direction.
  </Description>
 <Indexes>
    <Index id="first" value="intml.selection"/>
    <Index id="papers" value="liang93"/>
  </Indexes>
  <Filter id="select" type="Spotlight">
    <ShortDesc>Basic selection behavior</ShortDesc>
  </Filter>
  <Filter id="feedback" type="FeedbackSetIT">
    <ShortDesc>Visual feedback</ShortDesc>
  </Filter>

  <IPort id="posHead" type="Pos3D"></IPort>
  <IPort id="qHead" type="Quaternion"></IPort>
  <IPort id="posHand" type="Pos3D"></IPort>
  <IPort id="qHand" type="Quaternion"></IPort>
  <IPort id="a" type="float"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <IPort id="addObjectToScene" type="VRObject"></IPort>
  <IPort id="removeObjectToScene" type="VRObject"></IPort>
  <IPort id="type" type="String"></IPort>
  <IPort id="color" type="Color"></IPort>
  <OPort id="object" type="VRObject"></OPort>

  <Binding iE="select" iP="object" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="posHead" oE="select" oP="pHead" />
  <Binding iE="_self" iP="qHead" oE="select" oP="qHead" />
  <Binding iE="_self" iP="posHand" oE="select" oP="pHand" />
  <Binding iE="_self" iP="qHand" oE="select" oP="qHand" />
  <Binding iE="_self" iP="a" oE="select" oP="a" />
  <Binding iE="_self" iP="scene" oE="select" oP="scene" />
  <Binding iE="_self" iP="addObjectToScene" oE="select" oP="addObjectToScene" />
  <Binding iE="_self" iP="removeObjectToScene" oE="select" oP="removeObjectFromScene" />
  <Binding iE="_self" iP="type" oE="feedback" oP="type" />
  <Binding iE="_self" iP="color" oE="feedback" oP="color" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="select" iP="object" oE="_self" oP="object" />
</FilterClass>

<FilterClass id="SelectByScaleIT">
  <Filter id="select" type="SpotlightIT"></Filter>
  <Filter id="feedback" type="FeedbackScaleSetIT"></Filter>

  <IPort id="posHead" type="Pos3D"></IPort>
  <IPort id="qHead" type="Quaternion"></IPort>
  <IPort id="posHand" type="Pos3D"></IPort>
  <IPort id="qHand" type="Quaternion"></IPort>
  <IPort id="a" type="float"></IPort>
  <IPort id="scene" type="Scene"></IPort>
  <IPort id="addObjectToScene" type="VRObject"></IPort>
  <IPort id="removeObjectToScene" type="VRObject"></IPort>
  <IPort id="maxRadius" type="float"></IPort>
  <OPort id="object" type="Scene"></OPort>

  <Binding iE="select" iP="object" oE="feedback" oP="obj" />
  <Binding iE="_self" iP="posHead" oE="select" oP="posHead" />
  <Binding iE="_self" iP="posHead" oE="feedback" oP="pHead" />
  <Binding iE="_self" iP="qHead" oE="select" oP="qHead" />
  <Binding iE="_self" iP="qHead" oE="feedback" oP="qHead" />
  <Binding iE="_self" iP="posHand" oE="select" oP="posHand" />
  <Binding iE="_self" iP="qHand" oE="select" oP="qHand" />
  <Binding iE="_self" iP="a" oE="select" oP="a" />
  <Binding iE="_self" iP="scene" oE="select" oP="scene" />
  <Binding iE="_self" iP="addObjectToScene" oE="select" oP="addObjectToScene" />
  <Binding iE="_self" iP="removeObjectToScene" oE="select" oP="removeObjectToScene" />
  <Binding iE="_self" iP="maxRadius" oE="feedback" oP="maxRadius" />
  <Binding iE="select" iP="object" oE="_self" oP="object" />
</FilterClass>

<FilterClass id="RingMenu">
<ShortDesc>Basic behavior of a ring menu</ShortDesc>
  <Description>
A ring menu shows some options
arranged in a ring.The selected
option is the closest to the head
position. There is an object who
acts as a frame that receives
its coordinates from this object.
  </Description>
  <Indexes>
    <Index id="first" value="intml.control"/>
    <Index id="papers" value="liang93"/>
  </Indexes>
  <IPort id="posOffset" type="Pos3D" defValue="0, 0, 0">
     <ShortDesc>Starting position from the hand</ShortDesc>
  </IPort>
  <IPort id="qOffset" type="Quaternion" defValue="0, 0, 0, 1">
     <ShortDesc>Starting orientation from the hand</ShortDesc>
  </IPort>
  <IPort id="pHead" type="Pos3D" defValue="0, 0, 0">
     <ShortDesc>Viewpoint position</ShortDesc>
  </IPort>
  <IPort id="pHand" type="Pos3D" defValue="0, 0, 2">
     <ShortDesc>Hand position</ShortDesc>
  </IPort>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <IPort id="qHand" type="Quaternion" defValue="0, 0, 0, 1">
     <ShortDesc>hand orientation</ShortDesc>
  </IPort>
  <IPort id="setObjs" type="Scene">
     <ShortDesc>Objects in the menu</ShortDesc>
  </IPort>
  <OPort id="object" type="VRObject">
     <ShortDesc>selected object</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
     <ShortDesc>Frame position</ShortDesc>
  </OPort>
  <OPort id="q" type="Quaternion">
     <ShortDesc>Frame orientation</ShortDesc>
  </OPort>
  <OPort id="setFrameVisible" type="boolean">
     <ShortDesc>If the frame should be visible or not</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="RingMenuIT">
 <ShortDesc>Menu in a ring</ShortDesc>
  <Description>
Selection technique that
shows a ring of objects
to the user, with the closest
one as selected, and with an
interior frame for readability
purposes.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.control"/>
    <Index id="papers"
value="liang93"/>
  </Indexes>
  <Filter id="menu" type="RingMenu"></Filter>
  <Object id="frameObj" type="VRObject" primitive="Cylinder"></Object>
  <ObjectHolder id="frame" />

  <IPort id="posOffset" type="Pos3D"></IPort>
  <IPort id="qOffset" type="Quaternion"></IPort>
  <IPort id="posHand" type="Pos3D"></IPort>
  <IPort id="qHand" type="Quaternion"></IPort>
  <IPort id="setObjs" type="Scene"></IPort>
  <OPort id="object" type="VRObject"></OPort>

  <Binding iE="menu" iP="object" oE="_self" oP="object" />
  <Binding iE="menu" iP="pos" oE="frame" oP="setPos" />
  <Binding iE="menu" iP="q" oE="frame" oP="setQ" />
  <Binding iE="_self" iP="frameObj" oE="frame" oP="object" />
  <Binding iE="_self" iP="posOffset" oE="menu" oP="posOffset" />
  <Binding iE="_self" iP="qOffset" oE="menu" oP="qOffset" />
  <Binding iE="_self" iP="posHand" oE="menu" oP="pHand" />
  <Binding iE="_self" iP="qHand" oE="menu" oP="qHand" />
  <Binding iE="_self" iP="setObjs" oE="menu" oP="setObjs" />
  <Binding iE="menu" iP="object" oE="_self" oP="object" />
</FilterClass>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
</Package>
```

Interaction Techniques from Barrileaux

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Package PUBLIC "-//pfiguero//3D Interaction Techniques ML//EN"
    "intml.dtd">
<!-- Copyright info
  barrilleaux: InTs from his book "3D User Interfaces with Java3D"

  Copyright (C) 2001, Pablo Figueroa

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
  -->
<!-- 3D Interaction Techniques Markup Language -->
<!-- Version 0.01                              -->
<!-- Author: Pablo Figueroa                    -->
<!-- Contact Info: pfiguero@cs.ualberta.ca     -->
<!-- Changes:
  June 20 2001:
First version
  -->

<!-- Techniques described in Barrilleaux's    -->

<Package id="barrilleaux">
<Import id="objects"/>
<Import id="ualberta.appEnv.classes"/>

<ObjectClass id="DObject">
  <ShortDesc>Object in display space</ShortDesc>
  <Description>
      An object in display space. The display is the space that is in front
of the user's view. It contains layers numbered from 1 (the closest)
and above.
  </Description>
  <Indexes>
    <Index id="first" value="intml.Object"/>
    <Index id="space" value="display"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <IPort id="setPos" type="Pos2D">
    <ShortDesc>Changes the position of an object</ShortDesc>
    <Description>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        Changes the position of an object. It is considered a relative
        position if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="posChanged" type="Pos3D">
    <ShortDesc>Informs when the object moves</ShortDesc>
  </OPort>
  <IPort id="setRotation" type="float">
    <ShortDesc>Changes the rotation of an object</ShortDesc>
    <Description>
        Changes the rotation of an object from the horizon.
It is considered a relative
        rotation if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="rotChanged" type="float">
    <ShortDesc>Informs when the object rotates</ShortDesc>
  </OPort>
  <IPort id="setLayer" type="int">
    <ShortDesc>Changes the display's layer of an object.</ShortDesc>
    <Description>
        Changes the layer in the display of an object
    </Description>
  </IPort>
  <OPort id="layerChanged" type="int">
    <ShortDesc>Informs when the layer of the object changes.</ShortDesc>
  </OPort>
  <IPort id="setScale" type="Vector2">
    <ShortDesc>Changes the scale of an object</ShortDesc>
    <Description>
        Changes the size of an object. It is considered a relative
        size if it is contained in another object.
    </Description>
  </IPort>
  <OPort id="scaleChanged" type="Vector2">
    <ShortDesc>Informs when the object changes its size</ShortDesc>
  </OPort>
  <IPort id="setMatrix" type="Matrix3">
    <ShortDesc>Changes the rigid transformations.</ShortDesc>
    <Description>
        Changes the rigid transformations that apply to the object.
        It is useful when it necessary to apply rigid transformations
        at once and at a specific order.
    </Description>
  </IPort>
  <IPort id="addObject" type="VRObject">
    <ShortDesc>Adds a new part to this object</ShortDesc>
  </IPort>
  <OPort id="objectAdded" type="VRObject">
    <ShortDesc>Informs when a part is added to the object</ShortDesc>
  </OPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Removes a part from this object</ShortDesc>
  </IPort>
  <OPort id="objectRemoved" type="VRObject">
    <ShortDesc>Informs when a part is removed from the object</ShortDesc>
  </OPort>
  <IPort id="setBB" type="boolean">
    <ShortDesc>Defines if the bounding box is visible or not</ShortDesc>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  </IPort>
  <OPort id="BBChanged" type="boolean">
    <ShortDesc>Informs changes in the bounding box visibility</ShortDesc>
  </OPort>
  <IPort id="setColor" type="Color">
    <ShortDesc>Changes the main color of an object</ShortDesc>
  </IPort>
  <OPort id="colorChanged" type="Color">
    <ShortDesc>Informs when the color changes</ShortDesc>
  </OPort>
</ObjectClass>

<FilterClass id="WRM_MoveBasic">
  <Description>
    Implements world-relative movement in a plane.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.control.move.details"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <IPort id="startPos" type="Pos3D">
<ShortDesc>Initial position</ShortDesc>
<Description>
  Initial position, when the movement is activated.
</Description>
  </IPort>
  <IPort id="p" type="Pos2D" defValue="0, 0">
<ShortDesc>Input position for the movement</ShortDesc>
<Description>
  It saves the position when changes to true. Later on, it
  computes an offset with this start position and the current
  input position
</Description>
  </IPort>
  <IPort id="plane" type="Plane3D">
<ShortDesc>Plane of movement</ShortDesc>
  </IPort>
  <IPort id="switch" type="boolean">
<ShortDesc>activates and deactivates the filter</ShortDesc>
  </IPort>
  <IPort id="object" type="VRObject">
<ShortDesc>Object to be moved</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>New position for the object</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="Switch2Pos">
  <Description>
    Sends the last position when the input switch changes from
    false to true.
  </Description>
  <Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <Index id="first"
value="intml.control.move.details"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <IPort id="p" type="Pos2D" defValue="0, 0">
<ShortDesc>Input position for the movement</ShortDesc>
  </IPort>
  <IPort id="switch" type="boolean">
<ShortDesc>activates and deactivates the filter</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>Position when the switch changes to true</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="WRM_MoveOffset">
  <Description>
    Implements world-relative movement in a plane, with an offset from
    the starting position
  </Description>
  <Indexes>
    <Index id="first"
value="intml.control.move"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <Filter id="moveBasic" type="WRM_MoveBasic"></Filter>
  <Filter id="switch2Pos" type="Switch2Pos"></Filter>
  <ObjectHolder id="target" />
  <IPort id="p" type="Pos2D" defValue="0, 0">
<ShortDesc>Input position for the movement</ShortDesc>
  </IPort>
  <IPort id="plane" type="Plane3D">
<ShortDesc>Plane of movement</ShortDesc>
  </IPort>
  <IPort id="switch" type="boolean">
<ShortDesc>activates and deactivates the filter</ShortDesc>
  </IPort>
  <IPort id="object" type="VRObject">
<ShortDesc>Object to be moved</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>New position for the object</ShortDesc>
  </OPort>

  <Binding iE="_self" iP="pos" oE="moveBasic" oP="p" />
  <Binding iE="_self" iP="pos" oE="switch2Pos" oP="p" />
  <Binding iE="_self" iP="plane" oE="moveBasic" oP="plane" />
  <Binding iE="_self" iP="switch" oE="moveBasic" oP="switch" />
  <Binding iE="_self" iP="switch" oE="switch2Pos" oP="switch" />
  <Binding iE="_self" iP="object" oE="moveBasic" oP="object" />
  <Binding iE="_self" iP="object" oE="target" oP="object" />
  <Binding iE="_self" iP="viewpoint" oE="moveBasic" oP="viewpoint" />
  <Binding iE="moveBasic" iP="pos" oE="_self" oP="p" />
  <Binding iE="moveBasic" iP="pos" oE="target" oP="setPosition" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="switch2Pos" iP="pos" oE="moveBasic" oP="startPos" />
</FilterClass>

<FilterClass id="Viewpoint2Plane">
  <Description>
    Computes a plane perpendicular to the viewpoint at a given distance.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.control.move.details"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <IPort id="v" type="Viewpoint">
<ShortDesc>Viewpoint</ShortDesc>
  </IPort>
  <IPort id="d" type="float" defValue="0.0">
<ShortDesc>Distance from the viewpoint</ShortDesc>
  </IPort>
  <OPort id="plane" type="Plane2D">
<ShortDesc>Plane perpendicular to the viewpoint</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="DRM_MoveOffset">
  <Description>
    Implements display-relative movement in a plane, with an offset from
    the starting position
  </Description>
  <Indexes>
    <Index id="first"
value="intml.control.move"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <Filter id="moveBasic" type="WRM_MoveBasic"></Filter>
  <Filter id="switch2Pos" type="Switch2Pos"></Filter>
  <Filter id="v2Plane" type="Viewpoint2Plane"></Filter>
  <ObjectHolder id="target" />
  <IPort id="p" type="Pos2D" defValue="0, 0">
<ShortDesc>Input position for the movement</ShortDesc>
  </IPort>
  <IPort id="switch" type="boolean">
<ShortDesc>activates and deactivates the filter</ShortDesc>
  </IPort>
  <IPort id="object" type="VRObject">
<ShortDesc>Object to be moved</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>New position for the object</ShortDesc>
  </OPort>

  <Binding iE="_self" iP="pos" oE="moveBasic" oP="p" />
  <Binding iE="_self" iP="pos" oE="switch2Pos" oP="p" />
  <Binding iE="_self" iP="switch" oE="moveBasic" oP="switch" />
  <Binding iE="_self" iP="switch" oE="switch2Pos" oP="switch" />
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <Binding iE="_self" iP="object" oE="moveBasic" oP="object" />
  <Binding iE="_self" iP="object" oE="target" oP="object" />
  <Binding iE="_self" iP="viewpoint" oE="moveBasic" oP="viewpoint" />
  <Binding iE="_self" iP="viewpoint" oE="v2Plane" oP="v" />
  <Binding iE="moveBasic" iP="pos" oE="_self" oP="p" />
  <Binding iE="moveBasic" iP="pos" oE="target" oP="setPosition" />
  <Binding iE="switch2Pos" iP="pos" oE="moveBasic" oP="startPos" />
  <Binding iE="v2Plane" iP="plane" oE="moveBasic" oP="plane" />
</FilterClass>

<FilterClass id="OrbitMovement">
  <Description>
    Computes a new position and orientation around an object, given
    a starting position. It keeps the distance from the starting position
    to the center of the object.
  </Description>
  <Indexes>
    <Index id="first"
value="intml.navigation.details"/>
    <Index id="papers"
value="_hidden"/>
  </Indexes>
  <IPort id="obj" type="VRObject">
<ShortDesc>Object to orbit around to</ShortDesc>
<Description>
    Reference for the orbit
</Description>
  </IPort>
  <IPort id="alpha" type="float">
<ShortDesc>Horizontal angle</ShortDesc>
  </IPort>
  <IPort id="beta" type="float">
<ShortDesc>vertical angle</ShortDesc>
  </IPort>
  <IPort id="startPos" type="Pos3D">
<ShortDesc>Starting position</ShortDesc>
  </IPort>
  <IPort id="startQ" type="Quaternion">
<ShortDesc>Starting rotation</ShortDesc>
  </IPort>
  <OPort id="pos" type="Pos3D">
<ShortDesc>New position</ShortDesc>
  </OPort>
  <OPort id="q" type="Quaternion">
<ShortDesc>New orientation</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="OrbitCamera">
  <Description>
    Implements navigation by moving around an object
  </Description>
  <Indexes>
    <Index id="first"
value="intml.navigation"/>
    <Index id="papers"
value="barrilleaux01"/>
  </Indexes>
  <Filter id="orbitMovement" type="OrbitMovement"></Filter>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  <ObjectHolder id="v" />
  <IPort id="obj" type="VRObject">
<ShortDesc>Object to orbit around to</ShortDesc>
<Description>
    Reference for the orbit
</Description>
  </IPort>
  <IPort id="alpha" type="float">
<ShortDesc>Horizontal angle</ShortDesc>
  </IPort>
  <IPort id="beta" type="float">
<ShortDesc>vertical angle</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </IPort>

  <Binding iE="_self" iP="obj" oE="orbitMovement" oP="obj" />
  <Binding iE="_self" iP="alpha" oE="orbitMovement" oP="alpha" />
  <Binding iE="_self" iP="beta" oE="orbitMovement" oP="beta" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="viewpoint" iP="posChanged" oE="orbitMovement" oP="startPos" />
  <Binding iE="viewpoint" iP="qChanged" oE="orbitMovement" oP="startQ" />
  <Binding iE="orbitMovement" iP="pos" oE="v" oP="setPosition" />
  <Binding iE="orbitMovement" iP="q" oE="v" oP="setOrientation" />
</FilterClass>

<FilterClass id="PinocchioControl">
  <Description>
    Computes the new position for the 3rd person control and for the viewpoint.
  </Description>
  <Indexes>
    <Index id="first" value="intml.navigation.details"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <IPort id="pinocchio" type="DObject">
<ShortDesc>3rd person control</ShortDesc>
<Description>
    A control that represents the position and orientation of
    the viewpoint. It has two parts, body and nose, that control
    each of these parameters.
</Description>
  </IPort>
  <IPort id="dragControl" type="boolean">
<ShortDesc>inDragging</ShortDesc>
  </IPort>
  <IPort id="p" type="Pos2D">
<ShortDesc>input position for the movement</ShortDesc>
  </IPort>
  <IPort id="selObject" type="DObject">
<ShortDesc>selected part of pinocchio (body or nose)</ShortDesc>
  </IPort>
  <OPort id="posP" type="Pos2D">
<ShortDesc>New position for pinocchio</ShortDesc>
  </OPort>
  <OPort id="qP" type="float">
<ShortDesc>New orientation for pinocchio</ShortDesc>
  </OPort>
  <OPort id="pos" type="Pos3D">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<ShortDesc>New viewpoint position</ShortDesc>
  </OPort>
  <OPort id="q" type="Quaternion">
<ShortDesc>New viewpoint orientation</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="PinocchioCamera">
  <Description>
    Implements navigation by manipulating a camera representation, with a body
    for position and a big nose for orientation in a plane.
  </Description>
  <Indexes>
    <Index id="first" value="intml.navigation"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <Filter id="control" type="PinocchioControl"></Filter>
  <ObjectHolder id="rep" />
  <ObjectHolder id="v" />

  <IPort id="pinocchio" type="DObject">
<ShortDesc>3rd person control</ShortDesc>
<Description>
    A control that represents the position and orientation of
    the viewpoint. It has two parts, body and nose, that control
    each of these parameters.
</Description>
  </IPort>
  <IPort id="dragControl" type="boolean">
<ShortDesc>inDragging</ShortDesc>
  </IPort>
  <IPort id="p" type="Pos2D">
<ShortDesc>input position for the movement</ShortDesc>
  </IPort>
  <IPort id="selObject" type="DObject">
<ShortDesc>selected part of pinocchio (body or nose)</ShortDesc>
  </IPort>
  <IPort id="viewpoint" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </IPort>

  <Binding iE="_self" iP="pinocchio" oE="rep" oP="object" />
  <Binding iE="_self" iP="pinocchio" oE="control" oP="pinocchio" />
  <Binding iE="_self" iP="dragControl" oE="control" oP="dragControl" />
  <Binding iE="_self" iP="p" oE="control" oP="p" />
  <Binding iE="_self" iP="selObject" oE="control" oP="selObject" />
  <Binding iE="_self" iP="viewpoint" oE="v" oP="object" />
  <Binding iE="control" iP="posP" oE="rep" oP="setPosition" />
  <Binding iE="control" iP="qP" oE="rep" oP="setRotation" />
  <Binding iE="control" iP="pos" oE="v" oP="setPosition" />
  <Binding iE="control" iP="q" oE="v" oP="setOrientation" />
</FilterClass>

<ObjectClass id="ObjectBarrilleaux">
  <Description>
    Describes an object with extended features, in order to support InTs in
    Barrilleaux.
  </Description>
  <Indexes>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <Index id="first" value="intml.Object"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <Implements classId="VRObject"/>
  <IPort id="getObjectbyName" type="String">
<ShortDesc>sends the object with the given name</ShortDesc>
  </IPort>
  <OPort id="objectbyName" type="VRObject">
<ShortDesc>Response to getObjectbyName</ShortDesc>
  </OPort>
</ObjectClass>

<FilterClass id="CreateLabel">
  <Description>
    Creates a label given a string.
  </Description>
  <Indexes>
    <Index id="first" value="intml.manipulation.details"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>

  <IPort id="text" type="String">
<ShortDesc>Text for the label</ShortDesc>
  </IPort>
  <IPort id="axis" type="Vector3" defValue="0 1 0">
<ShortDesc>Axis to put the label, relative to the object</ShortDesc>
  </IPort>
  <IPort id="offset" type="float">
<ShortDesc>Offset from an axis</ShortDesc>
  </IPort>
  <OPort id="label" type="VRObject">
<ShortDesc>current viewpoint</ShortDesc>
  </OPort>
</FilterClass>

<FilterClass id="AddLabel">
  <Description>
    Adds a label to an object. It assumes the object doesn't have one.
  </Description>
  <Indexes>
    <Index id="first" value="intml.manipulation"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <ObjectHolder id="obj" />
  <Filter id="createLabel" type="CreateLabel"></Filter>

  <IPort id="text" type="String">
<ShortDesc>Text for the label</ShortDesc>
  </IPort>
  <IPort id="axis" type="Vector3" defValue="0 1 0">
<ShortDesc>Axis to put the label, relative to the object</ShortDesc>
  </IPort>
  <IPort id="offset" type="float">
<ShortDesc>Offset from an axis</ShortDesc>
  </IPort>
  <IPort id="object" type="VRObject">
<ShortDesc>Object that will contain the label</ShortDesc>
  </IPort>
  <OPort id="label" type="VRObject">
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<ShortDesc>current viewpoint</ShortDesc>
  </OPort>

  <Binding iE="_self" iP="text" oE="createLabel" oP="text" />
  <Binding iE="_self" iP="axis" oE="createLabel" oP="axis" />
  <Binding iE="_self" iP="offset" oE="createLabel" oP="offset" />
  <Binding iE="_self" iP="object" oE="obj" oP="object" />
  <Binding iE="createLabel" iP="label" oE="obj" oP="addObject" />
</FilterClass>

<FilterClass id="RemoveLabel">
  <Description>
    Removes a label from an object. It assumes the object has one object with
    name "label".
  </Description>
  <Indexes>
    <Index id="first" value="intml.manipulation"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <ObjectHolder id="obj" />
  <ObjectHolder id="obj2" />
  <Constant id="name" value="label"/>

  <IPort id="object" type="ObjectBarrilleaux">
<ShortDesc>Object that will contain the label</ShortDesc>
  </IPort>

  <Binding iE="_self" iP="object" oE="obj" oP="object" />
  <Binding iE="_self" iP="object" oE="obj2" oP="object" />
  <Binding iE="_self" iP="name" oE="obj" oP="getObjectbyName" />
  <Binding iE="obj" iP="objbyName" oE="obj" oP="removeObject" />
</FilterClass>

<ObjectClass id="ActivatebyTime">
  <Description>
    Turns on a variable for a certain period of time.
  </Description>
  <Indexes>
    <Index id="first" value="intml.manipulation.details"/>
    <Index id="papers" value="barrilleaux01"/>
  </Indexes>
  <IPort id="getObjectbyName" type="String">
<ShortDesc>sends the object with the given name</ShortDesc>
  </IPort>
  <OPort id="objectbyName" type="VRObject">
<ShortDesc>Response to getObjectbyName</ShortDesc>
  </OPort>
</ObjectClass>

</Package>
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix E

User Study Proposal to the Ethics Committee

The user study had to be approved by the Ethics committe at the University. This are the documents we presented for such approval.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Request for Ethical Review

Date November 4th, 2002

Researchers, Faculty Affiliation, Phone, Email

Walter F. Bischof
Associate Professor
492 3114
wfb@cs.ualberta.ca

Pierre Boulanger
Associate Professor
492 3031
pierreb@cs.ualberta.ca

Pablo Figueroa
PhD Student
492 7418
pfiguero@cs.ualberta.ca

Title of Proposal, Type (e.g., research project, grant application)

User Study on Retargetable Virtual Reality Interfaces. Research Project.

Short Summary of Project (50 words or less)

We are conducting comparative studies on the use of virtual reality hardware, when the task in each setup is exactly the same. Each participant will be exposed to a task in a particular hardware setup, and we will collect performance data and opinions from questionnaires. We will compare results between subjects exposed to different hardware setups, which are: a standard PC, a SMART Board, a PC with a joystick and a head mounted display, a PC with stereo screen and a space mouse, and our Visroom. The analysis of the data will allow us to infer user preferences and performance issues related to the technology described.

1. Describe the source of research participants or data. For participants not obtained through the 104/105 system indicating the manner in which participation will be solicited and the nature of any inducements or promises offered for participation.

We are planning to recruit volunteers by email, in the Computing Science Department or in other faculties of the University. The text of such an email is attached in Appendix 1.

2. Describe the procedure to be used.

a) Participants read the general introduction of the experiment (See Appendix 2).
b) Participants read an sign the informed consent form (Appendix 5). If they decide not to continue, they can live the experiment at any time.
c) Participants fill a form with general information about them (See Appendix 3).
d) Participants start the test by a directed example, in which they get familiar with the interface and the way it works.
e) Participants perform the task in one of the four implementations of the application: A standard PC, a PC with a Space Mouse, a SMART Board, or a head mounted display with a joystick. Participants can decide not to do the experiment at any time, and the information recollected won't be included in the study.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

f) At the end of the task, participants will be asked to fill a second form, with their opinions about the application and the experiment (See Appendix 4).

3. Describe how you will deal with the issues of informed consent and continuing voluntariness of participation in the proposed research.

Participants are required to sign an informed consent form and will be informed that they may leave the experiment at any time.

4. Describe how you will grant anonymity to participants and how responses will be kept confidential. If names or other identifying information are coded with data, describe how access to data is limited and safeguarded. Indicate who will have access. If appropriate, describe how consent is obtained from participants for exceptions to anonymity/confidentiality. If data are to be taken from existing sources, discuss the implications of pre-existing (implicit or explicit) guarantees of confidentiality/anonymity.

Data provided by each participant will be coded in such a way that anonymity of participants is guaranteed. Data will only be accessible to authorized personnel. Published data will contain no indication of the participants' identities.

5. If concealment and/or deception is to be employed, provide explicit justification. Indicate how and when participants will be informed of the concealment and/or deception.

We will not use concealment nor deception in our experiment.

6. Describe the nature of any risks to the physical or psychological well-being or integrity of participants that might arise from your procedures, and discuss your justifications, safeguards, and resolutions for these risks where appropriate.

In the event of sickness or dizziness on the part of the participant, the experiment will be terminated immediately.

7. Indicate when participants will be debriefed, and describe the nature and extent of debriefing.

Participants will be informed about the nature of the experiment beforehand and will be fully briefed about goals/result after the experiment. They will also be encouraged to ask questions. See appendixes for details.

8. Describe any apparatus, element of the physical environment, substance or other materials that could cause harm to a participant if a malfunction, misuse, accident, allergic reaction, or side-effect were to occur. If the participant comes into contact with a potentially hazardous apparatus or material, who will be responsible for checking for defects/malfunctions, and on what schedule will inspections be made? If participants come into contact with some substance that could cause harm, please document your safeguards.

We do not expose participants to hazardous apparatus or materials.

9. Describe qualifications of research personnel if special conditions exist within the research that could cause physical or psychological harm or if participants require special attention because of physical or psychological characteristics, or if made advisable by other exigencies.

N/A

10. Describe any potentially hazardous duties that will be required of research personnel, including physical, mental, or legal risks. Describe the safeguards you have implemented for your personnel.

N/A

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Ethics Statement

I have read the University Standards for the Protection of Human Research Participants (1999) and agree that the proposed research will be conducted in accordance with the guidelines and policies therein.

Signature of researcher ______________________________ Date ____________________________

Signature of researcher ______________________________ Date ____________________________

Signature of researcher ______________________________ Date ____________________________

Signature of Department Chair__________________________ Date ____________________________

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Call for Participants

The following email will be sent to the Computing Science Department, asking for participants in the experiment.

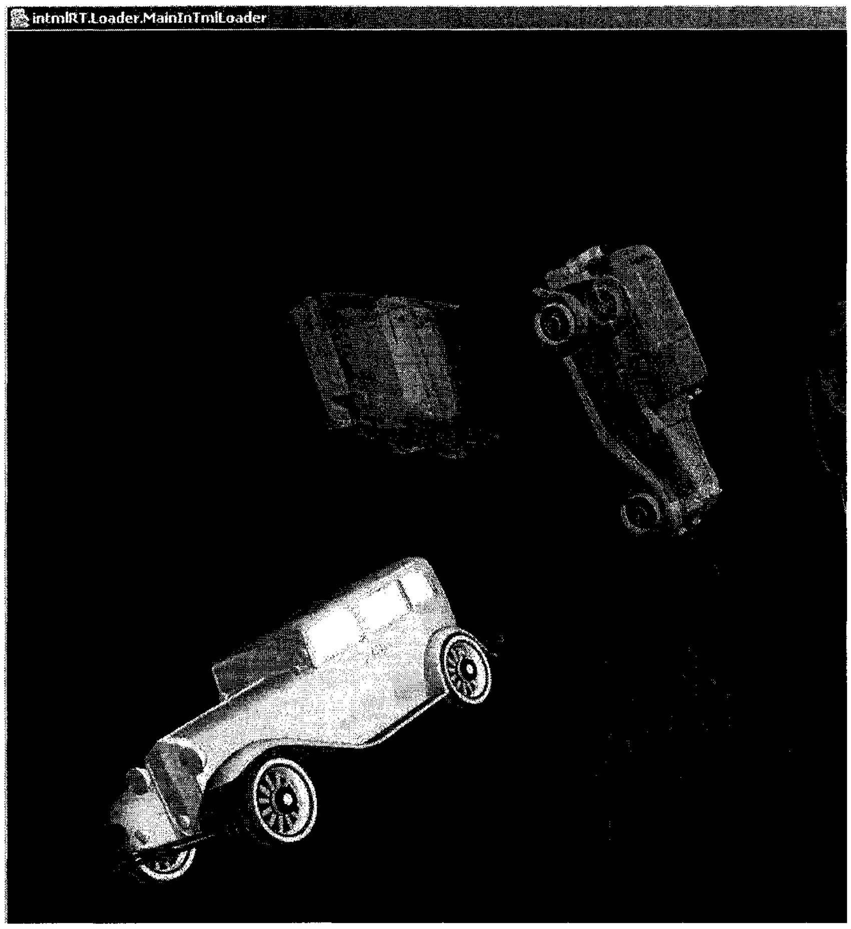
```
Subject: Call for volunteers in Virtual Reality

Hello,

We are conducting an experiment on interfaces for virtual reality
applications and we are interested in your participation. We will measure
your performance and your opinion about a simple task, moving and rotating
three graphical objects to a certain target position. Each participant
will perform this task for about 1 hour, in one of the hardware platforms
available in the Computer Graphics Lab. The data we will collect from your
participation will be anonymous, and will allow us to compare different
hardware setups in the virtual reality domain.

Thanks in advance for your collaboration. If you are interested in
volunteering please contact us by email to robyn@cs.ualberta.ca, with
your schedule preferences.
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A Matching Test

The purpose of this test is to compare different implementations of the same task in different hardware platforms. Tha task is a matching test, in which an object has to be moved to a target position. This particular test shows three objects: a red car, a yellow car, and a blue model [1] of Beethoven's face [2]. Each object has a semi-transparent copy, that defines the target position and orientation. In the application, you can select an object (not its copy), grab it, move it, and rotate it, until it matches its corresponding copy. Once an object and its copy are close enough, they dissapear. You can move the objects in any order, until all of them have been matched, in which moment the application ends. An example of how this application looks it is shown in the following figure.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



First, you will be asked to fill an introductory form with general information. After that we will ask you to match the three objects in a particular hardware platform, as soon as you can, and when you finish, we will ask you to fill a form with your opinions about the experience.

Thanks for your collaboration!

Special Considerations

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Each hardware platform has a different way to manipulate the objects, according to the particular input and output devices. The following links describe some details about each platform and the way tasks are accomplished.

- Standard PC
- SMART Board
- HMD and Joystick
- PC and SpaceMouse

People involved

Pablo Figueroa. PhD Student
Robyn Taylor. Research Assistant

References

[1] - Models provided as free of charge by Viewpoint.
[2] - Model provided as an example in the Java 3D distribution

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Introductory Questionnare

The following questions are general information about you and your involvement with computers.

Identification	Age	Gender: ○ M ○ F
	○ -20 ○ 21-25 ○ 26-30 ○ 31-35 ○ 35-40 ○ 41+	

Past Experience

How many operating systems (Varieties of Windows, Linux, Unix, Mac, ...) have you worked with?

○ none	○ 1	○ 2	○ 3-4	○ 5-6	○ more than 6

Of the following devices, software, and systems, check those that you have personally used:

Touch Screen ☐	Tracking devices [2] ☐	Voice Recognition ☐	☐
Joystick ☐	Stereo glasses [3] ☐	CAD (Computer Aided Design) ☐	☐
Track Ball ☐	Passive Stereo Displays [4] ☐	Video Editing Systems ☐	☐
Graphics tablet ☐	3D Mouse [5] ☐	Drawing/Painting Programs ☐	☐
Game Consoles ☐	☐	3D Video Games ☐	☐
SMART Boards [1] ☐	☐	2D Video Games ☐	☐
Head Mounted Display ☐	☐	Graphics Software ☐	
Pen Based Computer ☐	☐	☐	

Please write your comments (if any) about these list here:

Description

[1]. A SMART Board is a device produced by SMART Technologies for computer based presentations
[2]. A Tracking device allow the computer to know the position and orientation of the user's hand or head.
[3]. A stereo glasses device allows an user to see a special image in 3D.
[4]. A Passive stereo display allow users to see images in 3D without wearing special glasses.
[5]. A 3D mouse allow an user to move a pointer in 3D.

Immersive Tendency

Please circle the number that most appropiately reflect your impressions about using this computer system. Not Applicable = NA

How easily can you switch your attention from the task in which you are currently involved to a new task?	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easily	○ NA
How frequently do you get emotionally involved (angry, sad, or happy) in something you do?	Never	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA
How well do you feel today?	Not so well	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Excellent	○ NA
Are you easily distracted when working on a task?	Always	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Never	○ NA
How often do you play arcade or video games? (often is everyday)	Never	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Often	○ NA
How well do you concentrate on tasks you do not like?	Never	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA

Please write your comments about your immersive tendency here:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Questionnare About the Experience

The following questions are about your opinion of the experience you just had.

Identification [?]

Please circle the number that most appropiately reflect your impressions about using this computer system. Not Applicable = NA

Overall User Reactions

Overall reactions to the system	Terrible	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Wonderful	○ NA
	Frustrating	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Satisfying	○ NA
	Dull	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Stimulating	○ NA
	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
	Inadequate power	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Adequate power	○ NA
	Rigid	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Flexible	○ NA

Please write your general comments about the system here:

Screen

4.1a Objects in the screen	Hard to see	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy to see	○ NA
4.1.1 Quality of the image	Fuzzy	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Sharp	○ NA
4.3.1 Amount of information that can be displayed on screen	Inadequate	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Adequate	○ NA

Please write your comments about the screen here:

Learning

6.1 Learning to operate the system	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.1.1 Getting started	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.3.1Remembering specific rules about executing tasks	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.4 Tasks can be performed in a straightforward manner	Never	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA
Developing fine control of movement	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA

Please write your comments about learning here:

System Capabilities

7.1 System speed	Too slow	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Fast enough	○ NA

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

7.1.1 Response time for most operations	Too slow	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Fast enough	○ NA
7.2.2 System failures occur	Frequently	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Seldom	○ NA
7.4 Correcting your mistakes	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA

Please write your comments about system capabilities here:

Presence Items

How much were you able to control events?	Not too much	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Total control	○ NA
How responsive was the environment to actions that you initiated?	Not responsive	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Perfect response	○ NA
How natural did your interactions with the environment seem?	Not natural	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Natural	○ NA
How completely were all your senses engaged?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Completely	○ NA
How natural was the mechanism which controlled movement through the environment	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Natural	○ NA
How aware were you of your display and control devices?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very aware	○ NA
How much fatigue did you feel during your experience with the virtual environment?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	A lot	○ NA
How compelling was your sense of objects moving through the space?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very compelling	○ NA
Were you able to anticipate what would happen next in response to the actions that you performed	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA
How well could you move or manipulate objects in the virtual environment?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very well	○ NA
How involved were you in the virtual environment experience?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very involved	○ NA
How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very proficient	○ NA
How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?	Interfere a lot	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Not at all	○ NA
How much did the control devices interfere with the performance of assigned tasks or with other activities?	Interfere a lot	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Not at all	○ NA

Please write your comments about your sense of presence here:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Informed Consent Form

User Performance and Satisfaction in a Matching Test

Informed Consent Form

I, ______________________________, agree to participate in the study "User Performance and Satisfaction in a Matching Test" under the direction of Dr. Pierre Boulanger. I am aware that the application will record information about my performance during the task, for later analysis. I understand that my opinions in the questionnaires are anonymous and that my name will not be associated with the data. I understand that the use of these computers and virtual reality hardware might cause dizziness. I understand that the experiment takes approximately 1 hour to complete, and that I am free to discontinue participation at any time.

Signature: ___________________________ Date: ______________________________

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix F

A Brief Introduction to Z

This appendix gives a short introduction to the Z specification language and its notation, useful for understanding Section 3.5. A more complete description can be found in [87, 107].

The basic construct in Z is a schema. A schema defines both a type and a set of conditions. It can be used to describe a type, or a transformation over a type of values. The basic syntax of a schema is as follows

$$\begin{array}{|l}
\textit{SchemaName} \\
\hline
a1 : T1 \\
a2 : \mathrm{seq}(\mathrm{bag}\ T2) \\
\hline
\textit{conditions...} \\
\hline
\end{array}$$

In this example, a schema with name *SchemaName* is declared, as a set of variables and a set of conditions. Any instance of this schema will have two elements, *a1* or type *T1*, and *a2*, a sequence of bags of *T2*. *T1* and *T2* can be at the same time schemas by themselves, or basic types which are not described further. Conditions are predicates in propositional logic about the variables inside the schema, i.e. *a1* and *a2*.

A generic definition is a schema without a name, that defines functions that can be used in other schemas. A generic definition can also have parameters, in order to allow a more general definition of concepts. For example, from Section 3.5 we have:

$$\begin{array}{|l}
[X] \\
\hline
\_ \downarrow \_ : \mathrm{seq}\, X \times \mathbb{N} \longrightarrow \mathrm{seq}\, X \\
\hline
\forall\, st : \mathrm{seq}\, X;\ i : \mathbb{N} \bullet \\
st \downarrow i = (1 \mathinner{..} i) \lhd st \\
\hline
\end{array}$$

This defines the function $\downarrow$ with two arguments: a sequence of elements of type X (a schema parameter), and a natural number. The function maps these two values to a sequence of elements of type X. The condition part in this example has also some interesting notation. It can be read as follows: For any sequence st of elements of type X, and for any i natural number, $st \downarrow i$ corresponds to the domain restriction of st, for the elements between the natural numbers 1 and i.

A variable of type *SchemaName* can be declared as var : *SchemaName*. Variables inside var can be named as $var.a1$ or $var.a2$. $\mathbb{P}$ *SchemaName* corresponds to the power set (a set of variables) of variables of type SchemaName. θ*SchemaName* refers to a particular instance such a type.

Variables can be decorated in several ways, in order to add some semantics about their purpose. $var?$ refers to an input variable in a schema, while $var!$ is an output variable. var' refers to the state of var after certain operation, described by the schema that contains both variables.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Names for variables in a schema can be replaced, and variables can be hidden. For example *SchemaName*[*newName*/$a1$] is a schema identical to *SchemaName*, with the difference that all occurrences of $a1$ are replaced by *newName*. The schema *SchemaName* $\setminus (a1)$ is the following schema:

$$\begin{array}{|l}
\textit{SchemaName} \setminus (a1) \\
\hline
a2 : \mathrm{seq}(\mathrm{bag}\ T2) \\
\hline
\exists\, a1 : T1 \bullet \textit{conditions...} \\
\hline
\end{array}$$

A schema can be decorated by two special symbols: Δ and Ξ. Δ*Schema* is a schema that repeats twice the variables and constraints in *Schema*, once as they are, and a second copy decorated with $'$. Its purpose is to indicate an operation on *Schema*, a change in its state. Ξ*Schema* also defines both copies of variables, but also defines that both copies are the same. It refers to operations that does not change the state of *Schema*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

[1] 3D Connexion. Spacemouse plus. http://www.logicad3d.com/spacemouseplus.htm.

[2] Alias Wavefront. 3D Max. http://www.discreet.com/products/3dsmax/, 2003.

[3] Alias Wavefront. Maya. http://www.aliaswavefront.com/en/products/maya/index.shtml, 2003.

[4] Apache Foundation. Xalan website. http://xml.apache.org/xalanj/index.html.

[5] Avango: A distributed virtual reality framework. http://imk.gmd.de/docs/ww/ve/projects/proj1_2.mhtml, 2000.

[6] Ravin Balakrishnan and Ken Hinckley. The role of kinesthetic reference frames in two-handed input performance. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 171–178. ACM Press, 1999.

[7] Ravin Balakrishnan and Gordon Kurtenbach. Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 56–62. ACM Press, 1999.

[8] Jon Barrileaux. *3D User Interfaces With Java 3D.* Manning Publications, August 2000.

[9] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Addison-Wesley, 2nd edition, 2003.

[10] Shuvra S. Battacharyya, Praveen K. Murthy, and Edward A. Lee. *Software Synthesis from Dataflow Graphs.* Kluwer Academic Publishers, 1996.

[11] Steve Benford, Chris Greenhalgh, and David Lloyd. Crowded collaborative virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 59–66. ACM Press, 1997.

[12] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *Proceedings of IEEE Virtual Reality*, pages 89–96, 2001.

[13] Roland Blach, Jürgen Landauer, Angela Rösch, and Andreas Simon. A highly flexible virtual reality system. *Future Generation Computer Systems*, 14(3–4):167–178, 1998.

[14] Blender.org. Blender. http://www.blender.org/, 2003.

[15] I. M. Boier-Martin. Adaptive graphics. *Computer Graphics and Applications*, 23(1):6–10, January 2003.

[16] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 35–ff. ACM Press, 1997.

[17] Doug A. Bowman and Larry F. Hodges. Toolsets for the development of highly interactive and information-rich environments. *The International Journal of Virtual Reality*, 3(2):12–20, 1997.

[18] Doug A. Bowman and Larry F. Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *The Journal of Visual Languages and Computing*, 10(1):37–53, Februrary 1999.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[19] Doug A. Bowman, D. Koller, and Larry F. Hodges. A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual Reality*, pages 120–131, 1998.

[20] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture. A System of Patterns.* John Wiley & Sons, 1996.

[21] Rikk Carey and Gavin Bell. *The Annotated Vrml 2.0 Reference Manual.* Addison-Wesley, 1997.

[22] J. P. Chin, V. A. Diehl, and L. K. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218. ACM Press, 1988.

[23] Craig Cleaveland. *Program Generators with XML and Java.* Prentice Hall, 2001.

[24] Christian S. Collberg. Reverse interpretation + mutation analysis = automatic retargeting. In ACM, editor, *PLDI*, pages 57–70, 1997.

[25] 3D Connexion. Space ball. product description. http://www.3dconnexion.com/spaceball5000.htm, 2003.

[26] W3 Consortium. Extensible markup language *xml* 1.0 *secondedition.* http://www.w3.org/TR/2000/ REC-xml-20001006, October 2000.

[27] W3C Consortium. Validator for xml schema. http://www.w3.org/2001/03/webdata/xsv, 2002.

[28] World Wide Web Consortium. W3c xml pointer, xml base and xml linking. http://www.w3.org/XML/Linking, 2000.

[29] Bredemeyer Consulting. Motivating software architecture. http://www.bredemeyer.com/why.htm, 2001.

[30] Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, and Kevin Christiansen. Alice: lessons learned from building a 3d system for novices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 486–493. ACM Press, 2000.

[31] Microsoft Corporation. Sidewinder force feedback 2 joystick. http://www.microsoft.com/hardware/sidewinder/FFB2.asp.

[32] Mike Craven, Ian Taylor, Adam Drozd, Jim Purbrick, Chris Greenhalgh, Steve Benford, Mike Fraser, John Bowers, Kai-Mikael Jää-Aro, Bernd Lintermann, and Michael Hoch. Exploiting interactivity, influence, space and time to explore non-linear drama in virtual worlds. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 30–37. ACM Press, 2001.

[33] The cubes example in vr juggler 1.0.5. share/samples/ogl/cubes, 2000.

[34] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming: Methods, Tools, and Applications.* Addison–Wesley, 2000.

[35] Raimund Dachselt, Michael Hinz, and Klaus Meiner. Contigra: an xml-based architecture for component-oriented 3d applications. In *Proceeding of the seventh international conference on 3D Web technology*, pages 155–163. ACM Press, 2002.

[36] Raimund Dachselt and Enrico Rukzio. Behavior3d: an xml-based framework for 3d graphics behavior. In *Proceeding of the eighth international conference on 3D web technology*, pages 101–ff. ACM Press, 2003.

[37] Leonard Daly. Vrml2x3d.x3d. http://realism.com/x3d/presentations/Web3D-2002/worlds/Example/vrml2.x3d, 2002.

[38] DecisionSoft. Xml schema validator. http://tools.decisionsoft.com/schemaValidate.html.

[39] The dive home page. http://www.sics.se/dive/dive.html, 1991.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[40] Dale Dougherty and Arnold Robbins. *Sed & Awk*. O'Reilly & Associates, 2nd edition, 1997.

[41] Ralf Drner and Paul Grimm. Customizable interactions in 3d web applications with meta beans. In *Proceedings of the sixth international conference on 3D Web technology*, pages 127–134. ACM Press, 2001.

[42] Matthew Eldridge, Homan Igehy, and Pat Hanrahan. Pomegranate: a fully scalable graphics architecture. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 443–454. ACM Press/Addison-Wesley Publishing Co., 2000.

[43] M. Foskey, M.A. Otaduy, and M.C. Lin. Artnova: touch-enabled 3d model design. In *Virtual Reality, 2002. Proceedings. IEEE*, pages 119–126. IEEE, 2002.

[44] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[45] Enrico Gobbetti, Jean-Francis Balaguer, and Daniel Thalmann. Vb2: an architecture for interaction in synthetic worlds. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 167–178. ACM Press, 1993.

[46] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Passive real-world interface props for neurosurgical visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 452–458. ACM Press, 1994.

[47] Ken Hinckley, Joe Tullio, Randy Pausch, Dennis Proffitt, and Neal Kassell. Usability analysis of 3d rotation techniques. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 1–10. ACM Press, 1997.

[48] P.A. Howarth and M. Finch. The nauseogenicity of two methods of navigating within virtual interfaces. *Applied Ergonimics*, 30:39–45, 1999.

[49] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordan Stoll, Matthew Everett, and Pat Hanrahan. Wiregl: a scalable graphics system for clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 129–140. ACM Press, 2001.

[50] Virtual I-O. Virtual i-o glasses, 1995.

[51] VRCO Inc. Cavelib user's manual. http://www.vrco.com/CAVE_USER/index.html.

[52] Carnegie Mellon Software Engineering Institute. Why is software architecture important? http://www.sei.cmu.edu/publications/ documents/96.reports/ 96tr003/96tr0032.htm, 2003.

[53] Robert J. K. Jacob, Leonidas Deligiannidis, and Stephen Morrison. A software model and specification language for non-wimp user interfaces. *Transactions on Computer Human Interaction*, 6(1):1–46, March 1999.

[54] Hua Jiang, G. Drew Kessler, and Jean Nonnemaker. Demis: a dynamic event model for interactive systems. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 97–104. ACM Press, 2002.

[55] Daryl Kulak and Eamonn Guiney. *Use Cases: Requirements in Context*. Addison Wesley, 2nd edition, 2003.

[56] Yu-Kwong Kwok and Ishfaq Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys (CSUR)*, 31(4):406–471, 1999.

[57] Edward A. Lee and Thomas M. Parks. Dataflow process networks. In *Proceedings of the IEEE*, pages 773–799, May 1995.

[58] Jiandong Liang and Mark Green. Geometric modeling using six degrees of freedom input devices. In *3rd International Conference on CAD and Computer Graphics*, pages 217–222, 1993.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[59] Blair MacIntyre and Steven Feiner. Language-level support for exploratory programming of distributed virtual environments. In *UIST*, pages 83–94. ACM, 1996.

[60] Andrea H. Mason, Masuma A. Walji, Elaine J. Lee, and Christine L. MacKenzie. Reaching movements to augmented and graphic objects in virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 426–433. ACM Press, 2001.

[61] M. Massink, D. J. Duke, and S. Smith. Towards hybrid interface specification for virtual environments. In D. J. Duke and A. Puerta, editors, *Design, Specification and Verification of Interactive Systems '99*, pages 30–51, Wien, 1999. Springer-Verlag.

[62] Deborah J. Mayhew. *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interfac.* Morgan Kaufmann, 1999.

[63] Sun Microsystems. Java 3D Home Page. http://java.sun.com/products/ java-media/3D/index.html, 1997.

[64] Steven Molnar, John Eyles, and John Poulton. Pixelflow: high-speed rendering using image composition. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 231–240. ACM Press, 1992.

[65] Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40. ACM Press, 2002.

[66] Satoshi Nishimura and Tosiyasu L. Kunii. Vc-1: a scalable graphics computer with virtual local frame buffers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 365–372. ACM Press, 1996.

[67] Manuel Oliveira, Jon Crowcroft, and Mel Slater. Component framework infrastructure for virtual environments. In *Proceedings of the third international conference on Collaborative virtual environments*, pages 139–146. ACM Press, 2000.

[68] Jan Philipps and Bernhard Rumpe. Refinement of pipe-and-filter architectures. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 – Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing System. LNCS 1708, pages 96-115.* Springer, 1999.

[69] Jeffrey S. Pierce and Randy Pausch. Comparing Voodoo Dolls and HOMER: Exploring the Importance of Feedback in Virtual Environments. In *Proceedings of the CHI Conference*, pages 105–112. ACM, 2002.

[70] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. In *Eurographics.* Blackwell Publishers, 1998.

[71] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM Press, 1996.

[72] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. A framework and testbed for studying manipulation techniques for immersive vr. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 21–28. ACM Press, 1997.

[73] The matrix and quaternion faq. http://skal.planet-d.net/demo/matrixfaq.htm.

[74] Don Roberts and Ralph Johnson. Patterns for evolving frameworks. In Dirk Riehle Robert Martin and Frank Buschmann, editors, *Pattern Languages of Program Design 3.* Addison-Wesley, 1998.

[75] II Russell M. Taylor, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61. ACM Press, 2001.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[76] SensAble Technologies. Products: Phantom and ghost. http://www.sensable.com/products/ phantom_ghost/phantom.asp, 2003.

[77] Sense8. Worldtoolkit release 8 technical overview. http://www.sense8.com/products/wtk_tech.pdf, February 1998.

[78] Sense8. Virtual reality development tools. The sense8 product line. http://www.sense8.com/products/index.html, 2000.

[79] SGI. Iris performer home page. http://www.sgi.com/software/performer, 2003.

[80] Ehud Sharlin, Pablo Figueroa, Mark Green, and Benjamin Watson. A wireless, inexpensive optical tracker for the cave. In *Virtual Reality*. IEEE, 2000.

[81] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–328. ACM Press, 1992.

[82] Ben Shneiderman. *Designing the user interface: Strategies for effective human-computer interaction*. Addison–Wesley, 3rd edition, 1998.

[83] Mel Slater, Anthony Steed, John McCarthy, and Francesco Maringelli. The influence of body movement on subjcetive presence in viertual environments. *Human Factors*, 40(3):469–478, 1998.

[84] Shamus Smith, David Duke, and Meike Massink. The hybrid world of virtual environments. *Computer Graphics Forum*, 18(3):297–308, September 1999. ISSN 1067-7055.

[85] Shamus P. Smith and David J. Duke. Binding virtual environments to toolkit capabilities. *Computer Graphics Forum*, 19(3), August 2000.

[86] David N. Snowdon and Adrian J. West. The AVIARY VR System: A Prototype Implementation. In *6th ERCIM Workshop*, June 1994.

[87] Mike Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, 2nd edition edition, 1992.

[88] Kay M. Stanney, Mansooreh Mollaghasemi, Leah Reeves, Robert Breaux, and David A. Graeber. Usability engineering of virtual environments (VEs): Identifying multiple criteria that drive effective ve system design. *International Journal of Human Computer Studies*, 58(4):447–481, 2003.

[89] Anthony Steed and Mel Slater. A dataflow representation for defining behaviours within virtual environments. In *VRAIS*, pages 163–167. IEEE, 1996.

[90] Marc P. Stevens, Robert C. Zeleznik, and John F. Hughes. An architecture for an extensible 3d interface toolkit. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 59–67. ACM Press, 1994.

[91] Desney S. Tan, George G. Robertson, and Mary Czerwinski. Exploring 3D navigation: combining speed-coupled flying with orbiting. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 418–425. ACM Press, 2001.

[92] Vildan Tanriverdi and Robert J. K. Jacob. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press, 2000.

[93] Vildan Tanriverdi and Robert J.K. Jacob. Vrid: A design model and methodology for developing virtual reality interfaces. In ACM, editor, *Proceedings of the ACM Symposium of Virtual Reality Software and Technology*, pages 175–182. ACM Press, 2001.

[94] SMART Technologies. Smart board interactive whiteboard. http://www.smarttech.com/Products/smartboard/index.asp.

[95] D. Touraine, P. Bourdot, Y. Bellik, and L. Bolot. A framework to manage multimodal fusion of events for advanced interactions within virtual environments. In S. Müller W. St?rzlinger, editor, *Virtual Environments '02*, Eurographics, pages 159–168. Springer-Verlag Wien New York, 2002. Proc's Eurographics Workshop, Barcelona, Spain, 2002.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[96] H. Tramberend. Avocado: a distributed virtual reality framework. In *Virtual Reality, 1999. Proceedings.*, pages 14–21. IEEE, 1999.

[97] Carnegie Mellon University and University of Virginia. Alice: Easy interactive 3D graphics. http://www.alice.org, 1999.

[98] Scholarly Technology Group. Brown University. Xml validation form. http://www.stg.brown.edu/service/xmlvalid/.

[99] Martin Usoh, Ernest Catena, Sima Arman, and Mel Slater. Using presence questionnaires in reality. *Presence: Teleoperators & Virtual Environments*, 9(5):497–503, October 2000.

[100] VRCO. Cavelib library. http://www.vrco.com/products/cavelib/ cavelib.html, 2003.

[101] Colin Ware and Ravin Balakrishnan. Reaching for objects in vr displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(4):331–356, 1994.

[102] Colin Ware and Jeff Rose. Rotating virtual objects with real handles. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 6(2):162–180, 1999.

[103] Ken Watsen and Mike Zyda. Bamboo - a portable system for dynamically extensible, real-time, networked, virtual environments. In *Virtual Reality Annual International Symposium*, pages 252–259. IEEE, 1998.

[104] Web3D Consortium. Extensible 3D (X3D^{TM}) Graphics. Home Page. http://www.web3d.org/x3d.html, 2003.

[105] A.J. West, T.L.J. Howard, R.J. Hubbold, A.D. Murta, D.N. Snowdon, and D.A. Butler. Aviary – a generic virtual reality interface for real applications. In *Virtual Reality Systems*, May 1992.

[106] Bob Witmer and Michael J. Singer. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators & Virtual Environments*, 7(3):225–241, 1998.

[107] Jim Woodcock and Jim Davies. *Using Z. Specification, Refinement, and Proof.* Prentice Hall, 1996.

[108] XSLT: XSL Transformations. http://www.w3.org/TR/xslt11/.

[109] Shumin Zhai and Paul Milgram. Human performance evaluation of manipulation schemes in virtual environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 155–161, 1993.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.