University of Alberta

Enhancing Space Modeling and Mobile Resources Planning in Construction Operations Through A Simulation Driven Visualization Framework

by

Amr A. ElNimr

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

Doctor of Philosophy
in
Construction Engineering and Management

Department of Civil and Environmental Engineering

Fall 2011
Edmonton, Alberta

Dedication

To my mother Nazek whom without her efforts, love, sacrifice and dedication I wouldn’t be the man I am today. To my late dad; Dr. Abdel Khalek ElNimr; whose last words before I was leaving to Canada 5 years ago “only God knows if we will meet again or not”, and you were right dad we never met again. To my wife Maysoun who has been through a lot with me without even complaining, never thought one day that I will meet a lady with such strength, passion, kindness and forgiveness. To my baby Taliah, whom her laughter makes me feel how much God blessed me. To my friend whom I never met yet his image never escapes my memory; Al-Sayed Belal who was tortured to death by the Egyptian regime in January 2011. To the gentleman who is for me more than a professor or supervisor and to whom I have so much respect and admiration; Dr. Yasser Mohammed. To Dr.Simaan Abourizk, thanks for your support. To my friends Ahmed Taha, Ayman Shabana, Ismail Abdo, Mohammed Khalaf, Ahmed Nabih and Ahmed Elmekwad; you are the best guys. To my uncles and cousins.

Abstract

Simulation modeling is a strong tool that has not been utilized to its expected potential in day to day construction industry activities. One of the reasons contributing to that is the inability of simulation models to depict changes in site space in an intuitive way. This research tries to address this limitation in simulation modeling by developing a simulation driven visualization (SDV) framework which helps in modeling changes in both: site space geometry and site layout, throughout a simulated construction project. This framework makes use of the robustness inherited in simulation techniques and 3D modeling to provide a powerful tool that addresses the inability of simulation models to represent construction sites' spatial data in an intuitive way. The framework's pathfinding mechanism extension (PME) builds on the framework ability to model changes in space, through performing tempo-spatial planning for resources mobilization in a dynamically changing site layout throughout the lifecycle of a construction project. This is done through the site mesh generation mechanism and the A* search algorithm implemented inside the framework. Simulation driven visualization is a relatively new area of research in the construction management field. This area of research integrates Computer-Aided Design (CAD) modeling with animation and simulation techniques to create SDV frameworks or mechanisms that build on the strengths of each of those individual components. Although the prominent of those frameworks have contributed much to this area of research in the construction management field, they do have some limitations. To achieve the aforementioned aims, this developed framework had to address some of the technical limitations that exist in the current construction research state of the art SDV mechanisms. This research presents a new SDV framework which uses distributed simulation as its foundation. In addition to using this framework to achieve the aforementioned goals, the framework is also used to validate construction logic and

validate simulation models. The research also makes use of visual analytics and display design principles in the framework's visualization component conceptual design and the transformation of the simulation's object classes' statuses into visualization behaviors, respectively.

Acknowledgement

I would like to thank Dr. Yasser Mohammed for his efforts in this research. Also, I would like to thank Dr. Simaan Abourizk for his guidance and help during this research. I would like to thank Dr. Mohammed Al-Hussein for his guidance and support during this research. I acknowledge and appreciate the efforts of our research partners at PCL namely Mr. Rick Hermann, for participating in this research. Also, I acknowledge the efforts of Mr. Stephen Hague for helping in this research.

Table of Contents

List of Tables

List of Figures

List of Symbols/Formulas

$\boldsymbol{f}(\boldsymbol{N})$ A* evaluation function which represents estimated cost of the cheapest solution through node N

$\boldsymbol{g}(\boldsymbol{N})$ A* evaluation function: The cost to reach the current node N from start node.

$\boldsymbol{h}(\boldsymbol{N})$ A* evaluation function: Estimated cost of the cheapest path from node N to the goal node N_G

$\boldsymbol{h_{sld}}(\boldsymbol{N})$ Euclidian distances (straight line distance) are used inside the A* algorithm

$\boldsymbol{N}(\boldsymbol{x})$ Euclidian Distance Equation: X-Coordinate value of Node (N)

$\boldsymbol{N_G}(\boldsymbol{x})$ Euclidian Distance Equation: X-Coordinate value of Goal Node N_G

$\boldsymbol{N}(\boldsymbol{y})$ Euclidian Distance Equation: Y-Coordinate value of Node (N)

$\boldsymbol{N_G}(\boldsymbol{y})$ Euclidian Distance Equation: Y-Coordinate value of Goal Node N_G

$\boldsymbol{X_{min}}$ Mesh Generation Mechanism: minimum coordinates defining the borders of the mesh covering site spaces on one side along the x axes

$\boldsymbol{Y_{min}}$ Mesh Generation Mechanism: minimum coordinates defining the borders of the mesh covering site spaces on one side along the y axes

$\boldsymbol{X_{max}}$ Mesh Generation Mechanism: maximum coordinates defining the borders of the mesh covering site spaces on one side along the x axes

$\boldsymbol{Y_{max}}$ Mesh Generation Mechanism: maximum coordinates defining the borders of the mesh covering site spaces on one side along the y axes

$\boldsymbol{X_c}$ Mesh Generation Mechanism: define the center coordinates of the mesh covering site spaces in the direction parallel to the x axes

$\boldsymbol{Y_c}$ Mesh Generation Mechanism: define the center coordinates of the mesh covering site spaces in the direction parallel to the y axes

$\boldsymbol{X_l}$ Mesh Generation Mechanism: the site plot dimensions along the x axes of the mesh covering site spaces

$\boldsymbol{Y_l}$ Mesh Generation Mechanism: the site plot dimensions along the y axes of the mesh covering site spaces

$\boldsymbol{\alpha}$ Statistical significance

γ CDF of Log-Pearson 3 Distribution: ***γ*** is a continuous parameter

β CDF of Log-Pearson 3 Distribution: ***β*** is a continuous parameter such that ***$\beta \neq 0$***

α CDF of Log-Pearson 3 Distribution: is a continuous parameter where **$\alpha>0$**

K CDF of the Generalized Extreme Value Distribution: is continuous shape parameter

μ CDF of the Generalized Extreme Value Distribution: is continuous shape parameter

σ CDF of the Generalized Extreme Value Distribution: is continuous shape parameter such that (*$\sigma>0$)*

List of Abbreviations

2D	2-Dimensional
3D	3-Dimensional
4D	4-Dimensional
A* Algorithm	A-Star Algorithm
API	Application Programming Interface
BIM	Building Information Modeling
BGE	Blender Game Engine
CAD	Computer Aided Design
CDF	Cumulative Distribution Function
COTS	Commercial Off The Shelf
DCV	Dynamic Construction Visualizer
DES	Discrete Event Simulation
FOM	Federation Object Model
GA	Genetic Algorithm
GPS	Global Positioning System
HLA	High Level Architecture
HSV	HLA based framework for construction operations Simulation Visualization
IEEE	Institute of Electrical and Electronics Engineers
IPO	Interpolated Animation Curve (in Blender Game Engine)
K-S Test	Kolmogorov-Smirnov Test
P-P Plot	Probability-Probability Plot
PME	Pathfinding Mechanism Extension
RASS	Resources Admissible Site Space
RFSS	Resources Forbidden Site Space

RTI	Runtime Infrastructure
SDV	Simulation Driven Visualization
UDP	User Datagram Protocol
VITASCOPE	Visualization of Simulated Construction Operations

Chapter 1: Introduction

Simulation modelling is an effective approach for analysing construction operations, yet it is not widely used by construction practitioners. Simulation models are usually unable to represent site space in an intuitive way. This research is trying to address the problem of simulation models inability to model site space in an intuitive way through simulation driven visualization. Visualization in general, and particularly in construction projects, is a convenient and intuitive way of conveying project information among various project parties. Recently, construction management researchers have been investigating adding a visualization component to construction simulation models in order to make these models more intuitive and appealing to decision-makers. These researchers argue that enhancing visualization and spatial representation of construction operations in a simulation environment can improve the adoption of simulation techniques by the industry.

3D modeling has been utilized in the construction industry due to its ability to represent prototypes of construction processes and the changes in site geometry. This research is trying to provide a new and intuitive way for modeling site's spatial data and then building on this space modeling ability through solving the shortest safe travel paths on a dynamically changing site layout with changing geometries problem. The research is trying to do that through a new concurrent Simulation Driven Visualization (SDV) framework in which a 3D modeling component is integrated with a discrete event simulation (DES) engine. The framework, which is also developed through this research, is based on distributed simulation High Level Architecture (HLA) Institute of Electrical and Electronics Engineers (IEEE) standards and allows for hooking different visualization components to the simulation environment with a great deal of flexibility. The framework provides concurrent simulation visualization. It also provides two-way data communication between the simulation and visualization components for data transformation.

The HLA based framework for construction operations simulation visualization (HSV framework), utilizes 3D modeling strengths to model and depict future changes in both: site layout geometry and site space that take place throughout the project's simulation. The framework has a pathfinding mechanism extension which evaluates various site layout geometries in terms of the resources mobilization durations throughout the project's lifecycle simulation. This pathfinding mechanism also automates the costly and time consuming planning process of the mobilization (movement) of onsite mobile resources. The HSV framework integrates an A* search algorithms inside its visualization component. This algorithm assists the construction team in achieving the shortest, obstacle-free paths for moving mobile resources in a dynamically changing site layout geometry modeled by the framework. The framework's ability to model changes in site space that couldn't otherwise be addressed by mere simulation models, in addition to the framework's pathfinding mechanism extension, both components interoperate to help the decision-makers to plan ahead for the following at any stage of the project's lifecycle:

- Modeling and experimenting with various site layout scenarios.
- Testing existing resources' paths, clearing obstacles from the predefined resources' paths.
- Minimizing the expensive heavy lift resources mobilizations time.
- Accurately calculating the expected heavy lift resources mobilization durations to incorporate them in the project's schedule.

1.1 Problem Statement

This research is trying to address the problem of simulation models inability to model site space changes in an intuitive way through the development of a simulation driven visualization based framework which can help in modeling changes in site space geometry throughout the project's simulation run. It is worth mentioning that other techniques which are implemented to model changes in site space and perform dynamic site layout planning in current construction research have their own limitations. These limitations pertaining to the current dynamic site layout planning techniques which will be explained in detail in Chapter 2 can be summarized in the following points:

- Provide only snapshots in time at the beginning of each site layout geometrical change. This can be counter intuitive to the various decision-makers in the project.
- Do not model future changes in site spaces; they only try to optimize the site layout based on certain criteria.
- Do not link the changes in site layout to changes in mobile resources paths and do not take these paths in consideration while planning/modeling the dynamic changes in site layout.

- Does not convey a better understanding of the various layout geometries to the decision-making team.

The HSV framework can be classified as a tool that helps to model expected changes in site space while enhancing resources' paths planning rather than an ultimate tool for dynamic site layout planning. This distinction will be explained further in Chapter 2.

The limitations in the current state of the art Construction Operations' SDV research prevent the construction industry from benefiting from such strong tool to enhance any of the following aspects of construction projects: modeling changes in site space and dynamic site layout planning/modeling. More details on these limitations will be discussed in Chapter 2. This explains why this HSV framework is being developed rather than relying on any of the existing tools to achieve the research objectives mentioned earlier. The limitations that restrict the utilization of existing construction visualization mechanisms/frameworks in achieving the goals mentioned earlier can be summarized in the following points:

- The existing SDV frameworks do not represent changes in a construction site space (spatial data); this in turn does not make them a reliable tool to use in the construction site layout modeling.
- The existing construction visualization mechanisms/frameworks whether being schedule-based or simulation-based, concentrate on visualizing the final construction product development rather than simulating and visualizing the construction processes.
- The existing SDV mechanisms do not allow information to be sent back from the visualization to simulation. The simulation scenarios can't be evaluated or adjusted based on the visualization output (one way data flow only). This makes it complicated for a construction team to use the current SDV mechanisms to try various construction processes and site layout scenarios.
- The existing SDV mechanisms are tight coupled; this means that the simulation and visualization components of those mechanisms are only able to work with each other. This limits the ability of the existing frameworks to be implemented to plan for a wider spectrum of construction processes or projects.

1.2 Research Objectives

Earlier phases of construction projects lifecycles are usually clouded with many uncertainties. Simulation modeling is usually a robust way for addressing these uncertainties. It is useful for a project management team to have a tool at earlier phases of construction projects that could help the team achieve the following: interactively depicting the expected changes in site space geometry through the project duration, modeling future changes in site layout and planning heavy resources movement (mobilization) based on this dynamically changing site geometry. This helps to reach a more educated decision on the various aspects of the project. The objectives of this research can be summarized as:

- Enhancing spatial data representation in construction processes simulation.
- Extending simulation visualization to model mobile resources movement on construction site.

Achieving these objectives can lead to more expert adoption of simulation modeling in the construction industry.

1.3 Thesis outline

This research is trying to achieve the objectives mentioned earlier. This thesis starts by a critical literature review of site space modeling research and SDV research in construction. Generic visualization research and visualization research in construction is also discussed. Visual analytics and display design principles used in the conceptual design of the framework's visualization component are discussed. The HSV framework development and its technical aspects are then discussed together with the way the framework models changes in site space that take place during a project. The framework's Pathfinding Mechanism Extension (PME) is then explained with its interoperating A* pathfinding mechanism and mesh generation mechanism. Finally, the HSV framework's application to a real construction industry case to model expected site spatial changes, and plan heavy lift cranes mobilization in a dynamically changing site layout geometry is shown. Chapter 1 gives the reader a general idea of what this research is about, its objectives, and the thesis structure.

Chapter 2 is dedicated to discussing and critically reviewing the construction literature related to the objectives and tools used in this research. The chapter starts by a critical literature review on site space modeling and dynamic site layout planning research. The chapter then gives a thorough critical literature review of the SDV research done in the construction domain. This review of SDV research in the construction domain was necessary to explain the reasons behind choosing to develop a new HSV framework based on High Level Architecture (HLA) distributed simulation standards to solve the space modeling and resources pathfinding problem rather than using any of the existing state of the art SDV mechanisms. The background of visualization research in the construction domain and the conceptual design of the HSV framework's simulation visualization screen based on display design and visual analytics design principles is discussed. Some of these principles were utilized as guides to choose the various visualization behaviors which represent the various construction projects object classes' simulation states. These guides were utilized later in the case study as an attempt to standardize this transformation of simulation states into visualization behavior in the construction simulation visualization research.

Chapter 3 discusses the technical details of the HSV framework and its PME development. The chapter discusses the HLA distributed simulation standards that the framework's design is based on. Chapter 3 also explains how the concurrent DES visualization was achieved and describes the various components of the framework. The chapter also explains how the framework is able to model changes in site spaces throughout the project's lifecycle simulation and the framework's two-way communication ability. Also, the chapter discusses the search algorithms implemented inside the framework's PME. It is the first time that Search Algorithms are utilized in construction research. These algorithms build on the framework's ability to model changes in site space and are used to find the resources' shortest paths in a dynamically changing site layout where the information pertaining to these paths are sent to the running discrete event simulation component.

Chapter 4 is the case study application chapter of this thesis. The framework with its PME is utilized in a real construction project site (Ft. Saskatchewan oil upgrader site) to depict changes in the project's site space geometry throughout the project's lifecycle simulation. This tests the framework's ability to model changes in both: site space and various site layout geometries. The case study also shows how the framework's PME builds on the framework's ability to model changes in site space by solving the heavy lift cranes shortest safe pathfinding in a changing site geometry problem. The chapter also discusses the framework’s visualization screens conceptual design based on the visual analytics and display design principles discussed in Chapter 2. The chapter finally discusses the estimation of resources mobilizations durations and distances on shortest paths through stochastic modeling, instead of deterministically estimating them through the HSV framework’s pathfinding mechanism extension.

Finally, overall conclusive comments regarding this research’s ability to address the issue of modeling of site space in simulation and solving the pathfinding problem of resources in dynamically changing construction sites are discussed in Chapter 5. The contributions of this research to both: the construction industry and construction research are explained. Potential future research to enhance the automation of resources mobilization pathfinding planning in dynamically changing site layouts is suggested.

Chapter 2: Research Background: Site Space Modeling, Simulation Driven Visualization and General Visualization

2.1 Introduction

This research is trying to address the problem of simulation models inability to represent site space in an intuitive way. To do that a Simulation Driven Visualization (SDV) framework which can use its visualization modeling ability to depict and model changes in site space that occur due to construction processes simulation was developed. The framework use its ability to model site spaces to address various resource mobilization issues related to a resource's need of a certain space at a certain scheduled project time in order to perform its scheduled event. The framework's Pathfinding Mechanism Extension (PME) invests in the framework ability to model changes in site space in solving the resources pathfinding problem in dynamically changing site geometry.

This chapter is dedicated to discussing and critically reviewing the construction literature related to the objectives and tools used in this research. The chapter begins by critically reviewing construction research related to dynamic site space modeling. The limitations in this reviewed research are discussed. Researchers have used various methods to model changes in site space with time. Although these methods are different from this novel approach of using SDV to model site space changes, it is worth summarizing those research efforts to demonstrate the novelty of the approach followed here.

State of the art SDV research in the construction domain is then critically reviewed. Existing construction SDV mechanisms developed in recent construction research together with their shortcomings are reviewed. The limitations in the current SDV mechanisms that were developed through this research and addressed through developing a new SDV framework based on High Level Architecture (HLA) distributed simulation standards to utilize in solving the space modeling and resources pathfinding problem are discussed.

Finally, the literature related to both: generic and construction visualization researches is discussed. This part of the review will be focusing on explaining the qualitative guides from the literature that were utilized to conceptually design the visualization component of the HLA based framework for construction operations Simulation Visualization (HSV framework) that were used to design the visualization component of the framework used to solve the resources tempo-spatial management problem discussed in Chapter 4 (case study chapter). These design guides were utilized from the literature on visual analytics design and display design. The technical details pertaining to the development of the HSV framework and the framework's applications in construction research will be explained in chapter 3.

2.2 Dynamic Site Space Changes Modeling Methods and Mobile Equipment Motion Planning in Construction Research

Site layout planning is an important task that involves the positioning of temporary facilities and structures onsite to minimize the site layout costs and correspondingly overall project costs (El-Rayes and Said, 2009). Site layout planning can be categorized into static and dynamic categories. Static site layout planning models produce a single site layout that identifies static locations for a project's temporary facilities. Those facilities are not allowed to move over the project's lifecycle; accordingly, this type of layout planning does not consider changes in site space availability (El-Rayes and Said, 2009). Simulation has been one of the methodologies adopted for static layout modeling (Dawood and Marasini, 2002; Zhou et al., 2009).

Dynamic layout planning is a similar technique to what this research present methodology proposes. It involves the changes in a site's available space throughout the project duration (El-Rayes and Said, 2009). These changes usually occur due to moving of temporary facilities throughout the lifecycle of the project. The models that deal with dynamic site layout planning in construction are limited (El-Beltagi et al., 2004); among the most common of these are genetic algorithms (El-Beltagi et al., 2004), 4D modeling (Zhang et al., 2000; Ma et al., 2005), dynamic programming (El-Rayes and Said, 2009) and Computer Aided Design (Sadeghpour et al., 2006). It is worth mentioning that Sadeghpour et al. (2006) attempted to add a visualization component (through CAD modeling) to allow for better representation of site to the user.

Despite the contributions of previous research to modeling changes in site space over a project's duration, the following shortcomings can be noticed:

- Those models provide screenshots of different optimal site layouts for each stage of the project. These models adopt a chronological procedure to identify a local optimal solution for each of the identified stages of the project duration (El-Rayes and Said, 2009). In reference to models utilizing genetic algorithm (GA) and 4D modeling, El-Rayes and Said (2009) said that producing local optimal solutions for each identified stage of the project duration means that this layout's efficiency at later project stages is greatly affected by decisions taken in early layouts, which does not guarantee a global optimal solution. Also, these approaches may provide infeasible solutions when early-located facilities may cause insufficient space for future facilities (El-Rayes and Said, 2009). It is worth mention that Sadeghpour et al. (2005) tried to" tackle this "screenshots issue" by providing a set of consecutive layouts throughout the project. Although this research attempt showed a creative way of solving this problem, yet it lacked the concurrent visualization component to depict both: the changes in site space and the dynamicity of these layout changes.

- Both GA and 4D-based models study changes in site space that result from changes in locations/numbers of movable temporary facilities, according to the literal definition in Dynamic Site Layout Planning. These models ignore the changes in site space that happen as the building structure (fixed facility) itself evolves. For example, the footings excavation of a certain group of buildings starts at a later date in a residential complex project, so the site space where the future excavations will be can be utilized during early stages of the project. 4D-based dynamic site layout models inherit the limitation associated with 4D modeling; this means that these models depict changes in site space only on the basis of overall construction product progress. They do not consider the effects of site space changes on mobile resource movement throughout the project duration. Also, 4D-based models consider space changes inside the constructed product itself, for example the layout inside each constructed floor (Ma et al., 2005).

Another recent study employed dynamic programming in modeling dynamic changes in site layout (El-Rayes and Said, 2009). This research had a significant contribution in terms of trying to address the "Local Optimal site layout problem" mentioned earlier, by using dynamic programming epochs (rules) for producing an overall optimal site layout plan that takes into consideration changes in temporary facilities at future project dates. In spite of these advances, the model exhibits the following drawbacks:

- As is the case for models utilizing 4D modeling and GA, it takes into consideration only temporary facilities, without allowing for cases where structures or fixed facilities might cause changes in site space.

- It lacks an intuitive depiction of changes in site space with time and the effects of site space changes on resources. El-Rayes and Said (2009) commented on this shortcoming by suggesting their model can be expanded in future research to support 3D modeling of site layouts, detailed planning of resource travel paths, and considering the dynamic impact of material procurement decisions on site storage and layout planning.
- The model does not have an integrated layout cost calculation mechanism; in fact, it relies only on rate calculation for temporary facilities' movement based on Manhattan distance. This has two shortcomings:
    - The resources' shortest paths are not taken into account (especially if they carry expensive operating costs as in the case of heavy-lift mobile cranes). The effects of site space change on heavy mobile resource operation costs can be significant.
    - Manhattan distance is not necessarily the shortest distance that a temporary facility would take to move between two points onsite. This may not make the changes in measured moved distance of temporary facilities sensitive to changes in site layout. The overall minimum cost is affected by this parameter.
- Safety planning is not taken into consideration when planning the dynamic site layout using this model.

In recent research site space modeling was used to address the various resources (mainly equipment) site spatial needs (Gominuka and Sadeghpour, 2008). This research focused on classifying resources and setting their choice criteria based on their space needs in comparison to the modeled site spatial data. Another more relevant research was done by Albahnassi et al. (2009) relates space geometry to heavy mobile equipment motion (mobilization) planning. This research however concentrated more on finding general safe routes for equipment.

It is worth mentioning that static layout planning is not the focus of this research, as this research is about modeling changes in site space over the project duration using a SDV framework and not producing a snapshot in time of an optimal site layout.

Since it was decided to use SDV based framework in modeling changes in site space and solving the pathfinding problem in this research, the state of the art SDV research in the construction domain is now critically reviewed. The limitations in the current SDV research that were addressed through developing a new SDV framework based on High Level Architecture (HLA) distributed simulation standards, to utilize in solving the space modeling and resources pathfinding problem are discussed.

2.3 Simulation Driven Visualization (SDV) Mechanisms in the Construction Research

The tool that is employed to model changes in site space with time is an HLA-based SDV framework. Before exploring the development of this framework and its pathfinding mechanism, it is appropriate to look at the background of SDV in construction research. A critical literature review of SDV research in construction and the limitations of existing simulation visualization package, frameworks and mechanisms (both commercial and research-oriented) are now discussed.

Simulation is defined as the process of designing a model of a real system and conducting experiments with this model for the purpose of either understanding the behavior of the system or evaluating various strategies for the operation of the system (Shannon, 1975). The strength of simulation approaches emanates from their ability to examine various scenarios of the simulated system, rather than giving a mathematically optimum result as is the case in mathematical optimizations. Most researches stress the fact that simulation is not used to its maximum potential in the construction industry (Ioannou and Martinez, 1996; Kamat and Martinez, 2001; Huang and Halpin, 1994; Tucker et. al., 1998). There are two major reasons for this lack of use:

- Most simulation models show results in statistical and text formats, which causes simulation to be perceived by the construction industry as a “black box.”

- The current simulation models do not consider the transformation of space that result from the evolution of the constructed product (Kamat and Martinez, 2001).

There are SDV commercial, off the shelf (COTS) packages that exist in the market. These include Delmia’s Quest® and Brooks Software AutoMod®. However, these COTS are generally concentrated on manufacturing operations. They are usually unable to effectively handle the additional complications introduced by the changes in the geometry of the construction site as work progresses, and their use to model and animate construction operations requires a radical change in the model conceptualization and thought process followed by construction model developers (Oloufa, 1993 and Kamat and Martinez, 2008). Another visualization and CAD package that is used in the construction industry is CATIA®. Although powerful on the CAD side, it is limited on the simulation side of processes as it lacks the DES engine. CATIA can be a better solution for 4D modeling—final product development modeling—but not for simulation of construction processes.

The first construction simulation tools to adopt graphics did so in the early 80s (Rohrer, 2000). This was done through attempts to link construction simulation packages to animation to achieve a post-process playback of the simulation (Halpin and Woodhead, 1976; Kalk and Douglas, 1980; Paulson et al., 1983). The "Utopian Framework" for earth moving operations was a good example (Oloufa and Crandall, 1992). Another attempt to extract product information from a CAD design into a simulation was done through the PSE (Product-oriented Simulation Environment) (Xu and AbouRizk, 1999). Recent research in construction operations SDV involves the use of simulation trace files to invoke post-simulation animation (replay) of the simulation. This includes the Dynamic Construction Visualizer (DCV) and Visualization of Simulated Construction Operations (VITASCOPE) by Kamat and Martinez, 2001, Kamat and Martinez, 2004 and Kamat and Martinez, 2008 respectively.

The above simulation visualization packages (both commercial and research oriented) provide advanced visualization capabilities. However, they inherit some characteristics that limit their use in visualizing simulation behaviors of construction operations. Some of these characteristics include:

- Post-processing visualization: After a simulation is complete, visualization is produced as a replayed record of what took place during the simulation run. It does not allow the decision-maker to interact with the simulation. This causes the visualization component also to rely on simulation trace statements produced by the simulation engine exclusively for visualization purposes. Recent efforts by Rekapalli and Martinez, 2009 and Rekapalli and Martinez, 2011 has focused on achieving two-way communication and user interaction with the visualization component of a DES visualization.

- Tight coupling between simulation and visualization engines: Existing SDV mechanisms and COTS packages are usually tightly coupled with whatever simulation engine they serve. A particular visualization component will usually work perfectly with a particular simulation engine but will be very time consuming or knowledgably demanding to tailor it to work with other simulation engines. This leads to compromises on the strengths of both the simulation and visualization components. Also, this makes the two tightly coupled components strong in depicting certain applications and weak when it comes to depicting others. Finally this decreases the reusability these tightly coupled frameworks or COTS packages and limits their use in day to day construction operations.

- Compromising the power of the visualization components: Graphics usually demands specialized and high computer processing power. Running simulation models is also a demanding task on computer processors. Running the two in parallel significantly increases the demand on computer hardware and may be deemed infeasible in case of highly detailed simulation and visualization models. In addition, there have been significant advancements and optimization of software specially developed for handling graphics and visualization tasks. Capitalizing on these advancements requires the development of a mechanism to allow different specialized simulation and visualization components to interoperate without limiting or compromising the strengths of each.

Next, the pure visualization research in construction is reviewed. Also, the qualitative guides from the literature that were utilized to conceptually design the visualization component of the HSV framework that were used to design the visualization component of the framework used to solve the resources tempo-spatial management problem discussed in Chapter 4 (case study chapter).

2.4 Visualization in Construction Research and Conceptual Design of Visualization Based on Visual Analytics and Display Design Principles

Data analysis through visualization has its appeal in the construction industry. The representation of data in its visual format amplifies cognitive ability and reduces complex cognitive work (Card et. al, 1999; Russell and Udaipurwala, 2004). Researchers have proven that humans can derive an overview of information from data better and faster if it is presented in a suitable visual format other than numerical scripts or tables (Russell et al., 2009). Also, that visual information can be processed by the human perception system in parallel as opposed to serial processing for textural information (Russell et al., 2009). That is why using non-visual formats to display simulation results has contributed to lack of simulation techniques utilization in the industry. This will be discussed later.

Visualization in construction engineering research can fall into one of the following categories; construction processes simulation visualization, 4D modeling of construction product with time, abstract construction management data visualization, 3D CAD modeling or Building Information Modeling (BIM). This research is concerned more with the first of these categories (SDV), as it introduces a new simulation visualization framework based on HLA standards (HSV framework). The literature pertaining to this category (SDV) was critically reviewed earlier.

Visualization was generally added to construction research to assist the decision-makers in achieving faster insights and better understanding of information than representing in a tabular or text format. This research goes beyond that by making use of the visualization component of the HSV framework in solving both: the dynamic changes in site spaces modeling and heavy lift resources mobilization pathfinding problem. The visualization inside the framework does not exist merely to help the user understand the simulation; it is mainly a tool to evaluate the various simulation scenarios and aid the decision making process. Visual data exploration was also implemented in the design of the visualization component of the HSV framework allowing the user to explore the depicted simulation processes and changes in site layout while the simulation is running.

This section will start by reviewing the literature related to construction visualization research in general. This will be followed by explaining the qualitative guides that could be utilized to conceptually design the visualization component of the HSV framework. These guides were used to design the visualization component of the framework used to solve the resources tempo-spatial management problem discussed in Chapter 4 (case study chapter). These design guides were utilized from the literature on visual analytics design and display design. The technical details pertaining to the development of the HSV framework and the framework’s applications in construction research will be explained in chapter 3.

2.4.1 General Visualization research in construction

Visualization in construction is a way to analyze data. This can be referred to as Visual Analytics; which is the science of analytical reasoning facilitated by interactive visual interfaces (Russell et. al, 2009; Cook, 2005). Limited studies in the construction domain discussed the good design practices of visualization in the industry. Other visualization literature in the construction domain was focused on the technical aspects of visualization design, whether it is depicting simulation of construction operations (SDV) or the end product progress with project time (4D Modeling). From these studies, the ones pertaining to SDV were critically reviewed earlier. The focus of this section is the literature pertaining to generic visualization and visualization design in the construction domain.

A rich literature has been developed over many years concentrating on 4D modeling of the end product to be constructed and its progress over project duration. This is referred to as 4D visualization. This research includes Mckinney and Fischer, 1998, Hessom and Mahdjoubi, 2004, and other literature related to 4D modeling and Building Information Modeling (BIM). The difference between 4D modeling and SDV is explained by Kamat and Martinez, 2002. 4D modeling is related more to depicting the completion of the constructed product or artifact to be constructed. The visualization in 4D modeling is derived by the percentage completion of the project schedule. 4D modeling in construction does not get into the details of resources interactions and depiction of the construction processes itself. On the other hand, SDV depicts the various interactions that take place in the simulated construction processes. These interactions could be between various resources, resources and constructed product or resources and site spatial representation. This HSV framework depicts the last 2 type of interactions which are the essences of its use in resource mobilization planning in a dynamically changing site layout. It is also worth mentioning that the visualization in SDV is simulation based and not scheduling based. 4D modeling was also utilized to minimize the potential for design and construction errors in the construction product, to determine the most suitable construction sequence, and to monitor the project's progress (Dawood et al., 2003; Sirprasert et al., 2005).

Songer et al., 2004, focused on using visual analytics as a means to abstract construction management data and make it more informative to the decision-maker or what he calls: "solving data-rich-information poor problem". Song et al., 2005, proposed a model for displaying project's data on the 3D building model itself (Song et al., 2005). They used visual aids to assess project performance, while Zeb et al., 2008 used linear planning charts to assess the project's schedule quality. Zhang et al., 2009 used digital images with a building information system to automate the calculation of project progress. A comprehensive study pertaining to visualization design and visual analytics in the construction domain was Russell et al., 2009. Russell et al. discussed the principles of good visual analytic design which they borrowed from generic computer visualization research. The guides that Russell et al., 2009 set for visual analytics design were utilized in the conceptual design phases of the visualization component of the HSV framework used to solve the heavy lift cranes pathfinding problem in the PCL Ft. Saskatchewan case study discussed in Chapter 4. In that study, the authors proved a hypothesis that the application of visual analytics to construction management functions improves the various construction management processes, improves communication between the various project parties, and improves decision making (Russell et al., 2009).

The design of visualizations to act as a form of visual analytics in the construction industry is not an arbitrary process; however, there should be design criteria utilized to carry out the conceptual design of the visualization. Substantive research and design challenges must be addressed in formulating data representations and crafting them into visual representations (Russell et al., 2009). Although Russell et al. discuss these criteria in terms of construction management data analysis design, not an SDV design, yet still it proved useful when conceptually designing the HSV framework's visualization component. Russell et al., 2009, go on to state these criteria which will be explained further in the next section. The conceptual design of the visualization component of the HSV framework used to solve the heavy lift cranes pathfinding problem in the PCL Ft. Saskatchewan case study will be discussed in Chapter 4 (case study and implementation chapter).

Among the design criteria that were utilized to conceptually design this framework's visualization component are the 13 display design principles (Wickens and Holland, 2000; Boff et al., 1986) which are summarized by Wickens et al. in their textbook on human factors in Engineering (Wickens et al., 2004)[1]. These 13 generic principles are

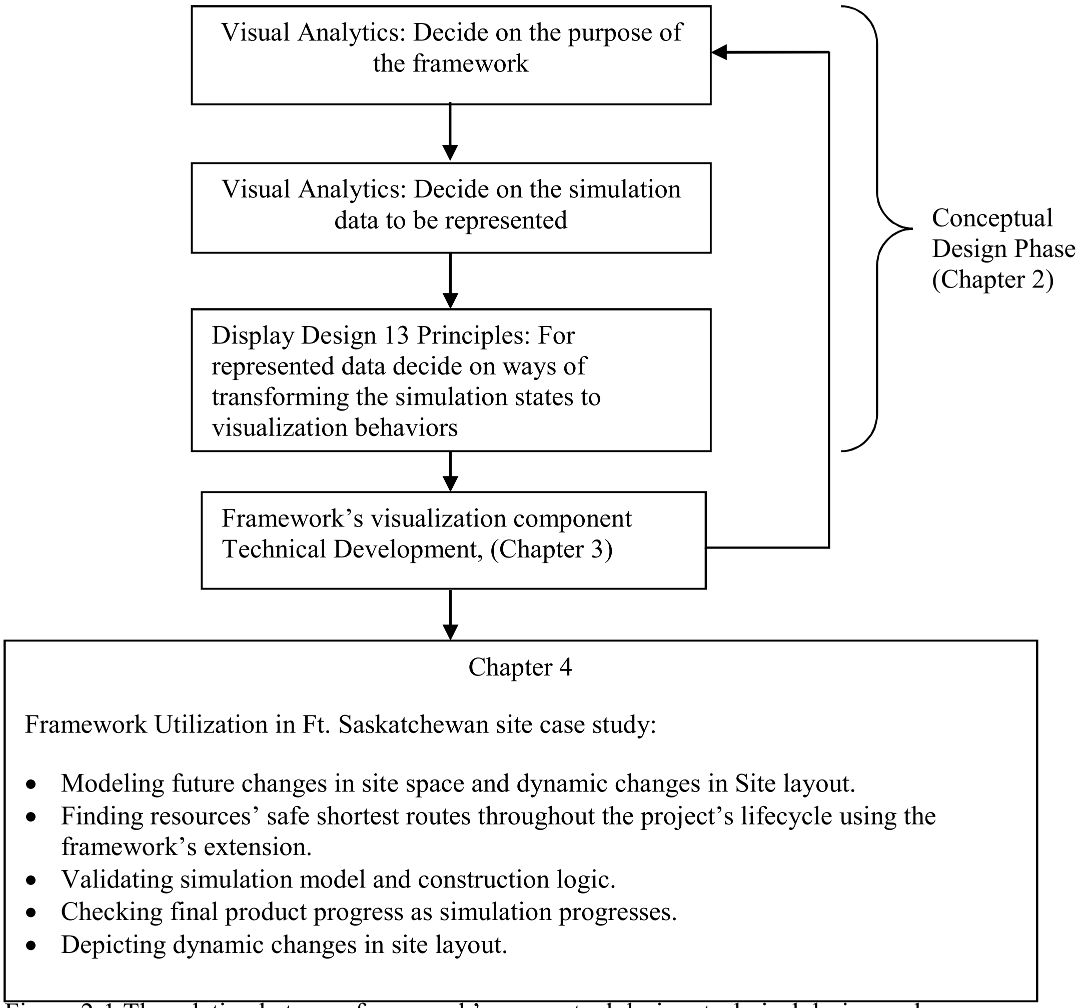
[1] For a comprehensive summary of the 13 principles of display design, please see "An Introduction to Human Factors Engineering" by Wickens et al., 2nd edition, 2004, published by Pearson Prentice Hall.

related to designing displays in general. The principles are related to strengths and weakness in human perception and humans' ability to process information. Although the visualization component of this HSV framework goes beyond displaying the simulated construction processes to also being an interface tool between the HSV framework and the decision-maker, the visualization still basically depicts the simulated construction processes. The decision-maker can use this interface tool to try different simulation scenarios and assess various site layouts to decide on shortest resource mobilization routes. This means that those display design principles can be applied to the design of this visualization. It is through the careful application of these principles to the output of information analysis that the best display emerges (Wickens et al., 2004). The principles will be explained further in the next section.

2.4.2 Visualization Conceptual Design Methodology

The HSV framework's visualization screens conceptual design together with the choice of visualization behaviors that depict the various construction elements' simulation states was based on certain qualitative display design guides. These guides are established in the literature of visual analytics and display designs. The design of visualizations is a new area in the construction research as mentioned earlier in Section 2.2. The development of the visualization component of the HSV framework was thought to be a good opportunity to utilize these guides in the construction management research. It was important to explain to the reader the basis on which the visualization component was conceptually developed before applying these guides for visualization screen design in Chapter 4 and before getting into the technical aspects of the framework's design in Chapter 3. The conceptual design of the visualization component of the HSV framework used to solve the heavy lift cranes pathfinding problem in the PCL Ft. Saskatchewan case study (Chapter 4) followed what is referred to as "the pillars of visual analytics effective display design" (Russell et al., 2009). Then, the 13 principles of human perception applied to display design (Wickens et al., 2004) were utilized in the translations of the chosen simulation's object classes attributes' states into corresponding visualization behaviors.

Figure 2-1 shows the iterative nature of conceptual design, the visualization literature design guides and principles used in this visualization component conceptual design process together with its relation to the framework's technical design and utilization. The figure also shows the logical link between the chapters of this thesis.


Visual Analytics: Decide on the purpose of the framework
Visual Analytics: Decide on the simulation data to be represented
Display Design 13 Principles: For represented data decide on ways of transforming the simulation states to visualization behaviors
Framework's visualization component Technical Development, (Chapter 3)
Conceptual Design Phase (Chapter 2)
Chapter 4

Framework Utilization in Ft. Saskatchewan site case study:

- Modeling future changes in site space and dynamic changes in Site layout.
- Finding resources' safe shortest routes throughout the project's lifecycle using the framework's extension.
- Validating simulation model and construction logic.
- Checking final product progress as simulation progresses.
- Depicting dynamic changes in site layout.

Figure 2-1 The relation between framework's conceptual design, technical design and framework's utilization, together with the chapters discussing these areas

2.4.2.1 Visualization Component: Conceptual Design In Light Of Visual Analytics Design Fundamentals

The term visual analytics environment refers to an information system that treats scenes of one or more visual representations and accompanying user interaction features that allows the decision-maker to interact with data and representations for analytic reasoning purposes (Russell et al., 2009). It is important to draw a line between the difference between the terms data visualization and visual analytics, where the former is considered only a part of the latter.

The main concern of designing such visualization for the construction engineering simulated process is the type of audience in the construction management decision-making field who will try to comprehend certain facts from the visually depicted processes. Heterogeneous user audiences with variable education backgrounds can exist in the same construction management team (Russell et al., 2009). Add to that the variations in the type of visual representations that these audiences are used to (for example 2D or 3D). Also, many construction data representation dimensions need to be translated to visual representation for a better depiction of the processes (Russell et al., 2009).

In addition to the above mentioned factors, an effective visual analytics design process is based on the following pillars (main steps) which were gleaned from a review of various visualization literatures by Russell et al., 2009:

- Understanding the purpose of design of visual analytics.
- Deciding on data to be represented (data representation) and the way to be represented (data transformation).
- Designing the representation and decision-maker interactions with the visualization.

These steps will be now explained before applying them to the conceptual design the visualization in the case study (Chapter 4). First the purpose of having the visualization should be known. Understanding the reasons behind the framework's visualization development is the key to which simulation object classes' states need to be represented in the visualization (data representation) and how they will be represented (data transformation). Once the data to be represented based on the visual analytics purpose has been chosen, it will be time to decide on the way that this simulation object classes' attribute values will be transformed into visual behaviors. For the remaining aspects of transforming the chosen represented simulation data to visualization behaviors, the 13 design principles of display design mentioned earlier will be utilized as guides for creating these visual behaviors. This will be explained in section 2.3.2 of this chapter.

2.4.2.2 Visualization Behaviors Choices In Light Of the 13 Display Design Principles

When the framework was utilized in the Ft. Saskatchewan site case study to solve the heavy lifting cranes pathfinding problem, (Chapter 4), the 13 display design principles were conceptually used as guides to map the various Discrete Event Simulation (DES) object classes' attributes states into corresponding visual behaviors. This section contains a briefing of those display design principles and the way they were utilized in the translation processes of the simulation's resources and products' attributes values into visual behaviors.

The HSV framework's simulation visualization can be classified along 3 different dimensions: the visualization physical properties, the tasks the visualization is designed to support and the framework's user properties (decision-makers' attributes). Physical properties of the HSV visualization screen include its classification as multi-coloured 3D visualization. The link between the visualization's display physical properties and achieving the tasks it was designed for can be built by what is known in the literature as the 13 display design principles. The link is made through building into the strengths of the decision-makers' perception, cognition and performance. The author thinks that the application of these principles in building the visualization component of this framework can act as a small initial step towards standardizing the simulation visualization developments in the construction engineering research.

The principles can be broken down into 4 categories:

A- Perceptual Principles:

1. The visualization should be legible or understandable and easy to comprehend by the decision-maker. Once the visualization design is legible, the remaining design principles that fall in the perceptual category can be applied to the design process. Figure 2-2 explains this conceptually.

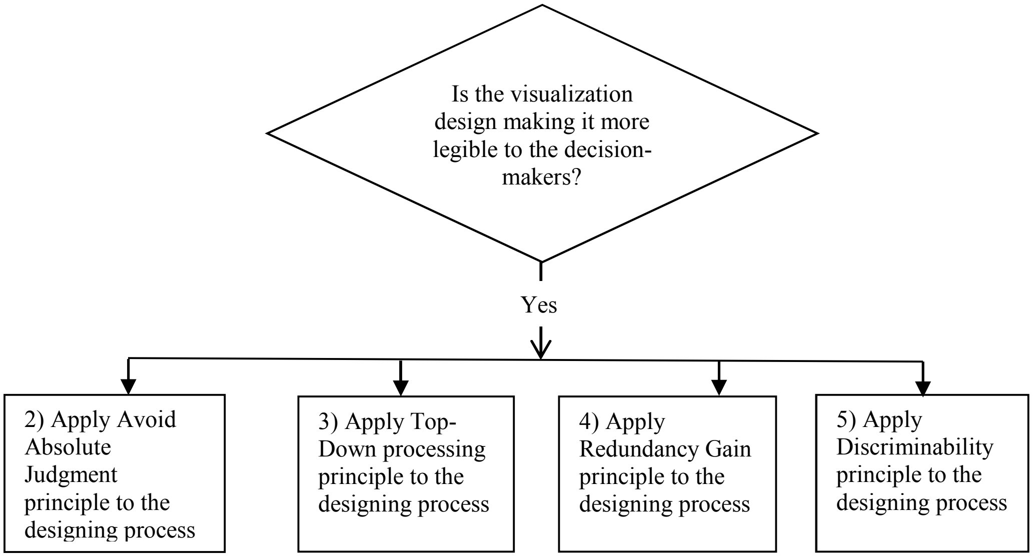

Is the visualization design making it more legible to the decision-makers?
Yes
2) Apply Avoid Absolute Judgment principle to the designing process
3) Apply Top-Down processing principle to the designing process
4) Apply Redundancy Gain principle to the designing process
5) Apply Discriminability principle to the designing process

Figure 2-2 Conceptual relation between the visualization perceptual design principles

2. The principle of avoiding absolute judgment limits states that nothing should bound the judgment of the decision-maker when interpreting a visualization behavior. For example, to require greater precision in color-coded map with nine hues is to invite errors of judgment.
3. Top-down processing or the principle of immediate context of a display is one other principle that was utilized when translating the simulation states into visualization behaviors. This implies that all the similar simulation events involving various object classes should be grouped and translated to similar visualization behaviors.
4. The translation also followed the principle of redundancy gain. For example, when the simulation triggers an event it should usually be mapped into two simultaneous visualization behaviors on the visualization display. This means that two different visualization behaviors represent a single simulation event. This is not considered a repetition but it is an application of the redundancy gain principle to stress the simulation event translation inside the decision-maker's mind.
5. Another principle used is the principle of discriminability. When different instances of the same object class have their simulation states translated into similar visualization behaviors, this can be confusing for the decision-maker.

B. Mental Model Principles:

6. Visualization components should support the principle of pictorial realism (Roscoe, 1968). The visualization should show a realistic picture of the simulation.

7. Inside the framework, moving Object Classes' 3D representations should follow the principle of moving parts. This principle indicates that inside a display, the elements that are involved in any type of movement should follow a certain spatial pattern that is compatible with the user's mental model of how this same element moves in the represented physical system. To get a clearer perspective on how this principle can be applied to construction operations SDV, one can say that; when a certain object is transformed during a construction process, either this movement is a

translation or rotation, the simulation of this process should have an object class that mimics how this object behaves in a real construction site. The visualization of this simulated process will in turn have a model that represents this object class, and the transformation(s) of this representation should correspond to its simulated movement.

C. Principles Based on Attention:

Simulation driven visualization falls within the category of a complex, multi-element display. Multi-element displays are characterized by the need for three different categories of attention: selective attention, focused attention and divided attention (Parasuraman et. al, 1984). The HSV framework can have multiple displays (which will be explained in the coming chapters). This means that a multiple simulated construction operation can be visualized simultaneously. Before getting into the application of these principles, it is worth explaining the various types of attentions that will be considered in the visualization display design:

- Selective Attention: When multiple simulated construction operations are depicted in a single or multi visualization screens in the HSV framework, the user can select to only concentrate on the visualization of a certain simulated construction operation.
- Focused Attention: The visual depiction of a selected simulated construction operation should be clear to the viewer without distractions from other construction processes being visualized.
- Divided Attention: This means that the HSV framework display design took in consideration the possibility that the decision-makers' judgement can also be based on divided attention between depicted simulated construction operations taking place on the same visualization screen.

The following 4 attention based principles were taken in consideration when designing the visualization screens of the HSV framework implemented in the Ft. Saskatchewan case study (Chapter 4) to solve the heavy lift cranes pathfinding problem. These 4 principles help to capitalize on the various categories of attention strengths mentioned earlier:

8. Minimizing the information access cost of the simulation visualization is another principle used in the visualization display design. There is typically a cost in time or effort for the decision-maker to move through multiple visualization screens in a single project's simulation. That is a cost to move the decision-maker's selective attention between different depicted simulated construction processes.

9. The HSV framework visualization's display design followed the principle of proximity compatibility. The application of this principle meant that if a simulation event is to be mapped into a visualization behavior, and this event involves interactions of more than one type of the simulation's object classes, then the 3D models representing those different object classes should be located in close proximity on the same display screen (visualization).

10. The principle of multiple resources: This principle implies that the visualization should be divided among various resources; for example, the use of audio plus visual formats to create certain visualization.

D. Memory principles:

The visualization of the simulated processes should always address the short and long term memory of the decision-makers. The last 3 display design principles will address this matter.

11. The principle of replacing memory with visual information (Norman, 1988) was utilized in the visualization design. In other words, the decision-maker using the HSV framework should not be left to rely on his/her memory to retrieve any data he/she uses to analyze a certain construction scenario being simulated and visualized.

12. The visualization should help the user in predicting the expected outcome of the simulated scenario. This is in accordance with the principle of predictive aiding (supporting proactive behavior). This helps the decision-maker using the framework to be proactive rather than reactive which adds more to the importance of the visualization component of the framework.

13. The visualizations inside the framework should be consistent. Hence, old habits from other displays (visualizations in our case) will transfer positively to support processing of the new displays (Wickens et. al; 2004).

2.4.3 General Comments

This section discussed the visual analytics and display design principles that will be used as guides in the conceptual design of the simulation visualization screens and simulation states translations into visualization behaviors. These guides were gathered from various visualization literatures. Although the visual analytics design guides were mainly used in the construction research to enhance construction management data visualization, they proved to be resourceful guides in the conceptual design process of the visualization screens in this framework. In addition, the 13 principles of display design which will be used in the mapping of simulation states into visualization behaviors inside the framework were defined. These principles were the output of a through literature review on the guides of computer visualization interface and display designs. In the following chapters, the technical details of the framework development, together with the framework's utilization to improve both: heavy lift resources mobilization planning and site space modeling in construction projects will be discussed. The implementation of the discussed visualization and display design principles in the conceptual design of the framework's visualization components will be discussed in the case study chapter (Chapter 4).

Chapter 3: Problem Solving Methodology, Framework and Pathfinding Mechanism Extension Developments

3.1 Introduction and Problem Definition

Simulation models fail to model site spaces in an intuitive way. On the other hand 3D modeling has proved to be a valuable tool when it comes to modeling geometry of shapes. In order to build on the simulation's modeling capabilities and 3D modeling's geometry depiction capabilities, and to overcome the limitations of simulation mentioned in Chapter 2, researchers turned to SDV as a tool that could better model the changes in site space. The shortcomings in current site space modeling research were discussed earlier in Chapter 2. SDV incorporates 2D and 3D depictions and animation to provide the end user with better representation and understanding of construction operations and space. Also, SDV addresses the limitations of using stand-alone animation or simulation systems where graphical representations of construction operations and simulation of activity timing and resource interactions are separate and independent of each other.

To overcome the limitations in state of the art construction operations SDV mechanisms that were developed through recent construction research (discussed in Chapter 2), a loosely coupled simulation visualization framework was developed (ElNimr and Mohamed, 2010; 2011). The HSV framework (ElNimr and Mohamed, 2010; 2011) is employed to model changes in site space with time and study their effects on resource paths onsite using a pathfinding mechanism implemented inside the visualization component.

This developed HSV framework does not produce single or multi-optimal site layouts, but simulates and depicts the changes in site space with time through each stage of the project. This is done by integrating a Discrete Event Simulation (DES) engine with a 3D engine in the developed SDV framework. The HSV framework depicts a changing site layout as at a real site. The framework takes models changes in site space due to both temporary facilities' and permanent structural elements' locations when depicting layouts and calculating shortest resource routes throughout the project's lifecycle.

The developed HSV framework provides an intuitive depiction of changes in site space throughout the entire project lifecycle. The heavy lift resources shortest safe pathfinding task performed by the framework's pathfinding extension is based on the changes in site space (changes in site layout). This is done through the framework's visualization component connected to the simulation model and the pathfinding mechanism developed inside this component for mobile resources pathfinding. Also, the HSV framework takes into consideration the safety factor when planning resources mobilizations given the changes in site space.

Based on that HSV framework can be a strong tool in modeling site space geometry. An application of modeling changes in site space geometry that result from dynamic changes in site layout, is modeling the changes in the geometry of mobile resources paths in a changing site layout.

Throughout a project's development, there are various objects with variant geometries occupying the spaces in a construction area. These objects can be; permanent structures (for example structural elements) which occupies more site space monotonically as the project time proceeds, or temporary site facilities which can be mobile and/or change their geometry as project time proceeds. All these objects cause changes in both; the site space geometry and mobile resources paths on site. Modeling a site's spatial data in an intuitive way inside simulation and then finding the heavy lift resources' shortest safe travel paths on a dynamically changing site layout with changing geometries is the problem this research is trying to address. This chapter explains the solution suggested by this research for this problem. The research presents a SDV framework based on distributed simulation High Level Architecture (HLA) standards with a Pathfinding Mechanism Extension (PME) to model site spaces geometry and find the shortest safe routes based on changes in these spaces. This chapter explains this generic framework and the methodological development steps for this framework.

The chapter starts by explaining and analyzing the general mobile resources pathfinding problem in a continually changing site layout and formulating it as an intelligent agent search problem. The chapter then explains the algorithms implemented in the PME of the framework to find the mobile objects shortest safe paths in dynamically changing site geometry. The choice of this pathfinding algorithm is also explained. Next, the SDV framework developed to depict simulated construction operations and model sites' spatial data geometries accordingly is explained. The development of this framework; HLA based Framework for Construction Operations Simulation Visualization (HSV); is then described. The pathfinding mechanism implemented in the framework's PME is then explained. The mechanism which is made up of 2 interoperable components; mesh generation mechanism and A* pathfinding algorithm finds the shortest safe routes in a continually changing site geometry based on changes in site's spatial data. Modeling of changes in site space based on construction processes simulation depictions is explained prior to the PME of the framework because modeling changes in site geometry and site space is a prerequisite to applying the pathfinding search mechanism as it will be explained later. The next chapter will showcase a comprehensive case study in which the HSV framework and its PME component are utilized to model changes in site space and plan resources mobilizations throughout the project's simulated lifecycle.

3.2 Search Algorithms and the A* Algorithm[2]

This section will discuss search algorithms, focusing on the A* algorithm implemented inside the HSV framework to solve the pathfinding problem in a site with changing geometrical layout. It will begin by discussing search problems, their formulation, and how the resource pathfinding problem in a dynamically changing site falls in this category of problems. Then the section will explain the reasons for choosing the A* search algorithm to use in the visualization component's PME of this framework in order to solve this search problem.

3.2.1 Formulating Construction Sites Pathfinding Problem as a Search Problem

Resources paths can generally be used for labor or heavy lift mobile resources mobilization. Expensive operating costs are usually associated with heavy lift mobile resources as it will be shown in Chapter 4 (case study chapter). In order to find these paths in a continually changing site geometry (dynamically changing site layout), there is a need for a solution that models future changes in site space throughout the project duration and then search for, and find the shortest safe mobile objects paths on the current site geometry, depending on site space availability.

Search in artificial intelligence refers to an object (agent) examining different possible sequences of actions that lead to states of known values, then choosing the best sequence based on the desired search criteria. A search algorithm returns a solution in the form of an action sequence. The first step in employing a search algorithm is to formulate this pathfinding problem as one that can be solved by implementing a search algorithm.

[2] Russell and Norvig's *Artificial Intelligence: A Modern Approach* 2nd Edition (2003), Chapters 2, 3 and 4 are the source of much of the information included in this section. For more information and derivation please check the aforementioned textbook.

The site resources' shortest path search problem can be formulated in terms of the following, as per search problems classification of Russell and Norvig (2003):

- Initial State: The object's initial location at the current site geometry. Can be expressed as "In<LocationID>". For example, if a resource is at Location 5 on the current site layout geometry, then its initial state is "In<5>".
- Goal State: The object's goal location. For example, if a resource needs to be moved from Location 5 to a certain lifting location, Location 9, the Goal State can be formulated as "In<9>".
- State Space: The set of all states reachable by an object from the initial state. For example, if a resource is at certain site location represented by certain coordinates, the State Space is all the other locations (states) that can be reached from this location. The State Space (reachable states) can be connected through a graph; this graph could be represented by a tree or grid. In our case the grid was chosen to cover modeled site spaces. The grid was chosen to cover the site layout (Figure 3-9) for the following reasons:
    - In this type of problem, a search algorithm has a high possibility of going through states (positions onsite) that have been already processed and considered not to fall on the shortest path between Initial and Goal States. This is called the "repeated states problem" (Russell and Norvig, 2003). Such a problem can cause a solvable search problem to become unsolvable if the algorithm cannot detect these repeated states. To avoid this problem, a graph-search algorithm rather than a tree-search algorithm is implemented. Graph-search algorithms (such as various forms of best-first search algorithms) are much more efficient than tree-search algorithms (such as breadth-first, uniform-cost and depth-first algorithms) in solving problems with expected many repeated searches (Russell and Norvig, 2003). A graph-search algorithm uses a grid rather than a tree to search the State Space.
    - A grid is the most intuitive way to cover the site area and accommodate for changes in site space geometry, as shown in Figure 3-9. Hence, each object path can be represented as a network of edges forming the grid. In other words, the nodes are the object's new locations (states), while the edges connecting them are the actions to reach those states. An algorithm

was implemented that utilizes nodes and edges to find the least-expensive, obstacle-free path onsite for mobile objects.
    - The grid (mesh) best models the objects' movement states, as it covers more possible mobile objects' states on the site space, allowing the object to move between any two states. The tree on the other hand would have limited the states that a resource can move on.
    - A grid is ideal to use with the mesh generation mechanism to depict changes in site space as will be explained later.
- Path Cost: The function that assigns cost to each object path. This will be the A* function, which will be explained in the next section. The Path Cost function in this case gives a distance, as the pathfinding mechanism is trying to obtain the shortest path for the movements of site's mobile objects.
- Actions: The set of actions that an object can take in order to move from one state to another before ultimately reaching the Goal State if a solution is found. For example, a resource needs to travel from one location to another taking the shortest alternative until reaching its intended lift location (Goal State).
- Goal Test: Tests if the object has reached its Goal State or not.

Now that the shortest resource paths search problem in a dynamically changing site layout is formulated in terms of the above elements, it is time to discuss the reasons behind the choice of the A* search algorithm to solve this problem.

3.2.2 The Choice of A* Search Algorithm: Reasons and Proof

The search algorithm used here is the A* algorithm. A* is a widely used graph-search algorithm, invented in 1968 by Peter Hart, Nils Nillson, and Bertram Raphael. It is an informed search algorithm, meaning that the algorithm can differentiate if any of the states of an object are better (more promising) than others in reaching the goal state.

The A* algorithm depends on an evaluation function $f(N)$ to decide on the mesh or graph node to expand to next. This $f(N)$ function measures the distance to the goal, and the lowest evaluation is selected for expansion. The $f(N)$ function is achieved by adding two functions: $g(N)$ and $h(N)$. The $g(N)$ function measures the exact distance from the start node to current processed node on a graph. The $h(N)$ is a heuristic function and a key component of this algorithm. It estimates the distance from the current node to the goal node on the site mesh. This heuristic function contributed to the choice of this algorithm rather than other search algorithms to solve the pathfinding problem. This will be explained in detail later in this section. Also, the A* algorithm takes into consideration the distance from the start node to the current node, in addition to the heuristic component of the equation as illustrated in Equation 1 below. The way in which this algorithm solves the resources' pathfinding problem inside the HSV framework will be explained in section 3.4.

The A* algorithm uses the following evaluation function:

$$f(N) = g(N) + h(N) \text{........ (1)}$$

Where,

N : Node being processed by the algorithm

$f(N)$: A* evaluation function that represents estimated cost of the shortest solution through node N.

$g(N)$: The cost to reach the current node N from start node.

$h(N)$: Estimated cost of the shortest path from node N to the goal node N_G

There are a number of search algorithms that find the shortest accessible path employing edges and nodes (either graph or tree search). The choice of the A* algorithm rather than other pathfinding algorithms was based on several research steps. The research steps will be analyzed in a systematic way to show how the decision to use the A* algorithm was ultimately reached:

- An informed search algorithm (such as A*) was preferred over an un-informed search algorithm (such as depth-first and breadth-first) for the following reasons:
    - Un-informed search algorithms (also called blind search) can't tell if any state of the searching object is more promising than another in terms of reaching the goal state (lifting location for resource).
    - Un-informed search strategies can be computationally expensive yet not reach an optimum solution. Russell and Norvig (2003) set out with 4 criteria to assess the various search algorithms: (1) Completeness—does the algorithm find a solution when there is one, (2) Optimality—does it find the optimal solution (the shortest path in our case), (3) Time complexity—how long does it take to find a solution in terms of

processing time, and (4) Space complexity—how much memory is needed to perform the search. The major types of un-informed search algorithms were evaluated based on these criteria.[3]Un-informed algorithms usually have high time and space complexities that can hinder (slow down) the communication and time management between the simulation and visualization components of the framework. The un-informed algorithms that might be low on time and space complexity are not guaranteed to be complete or optimal in finding the shortest path.

    - Informed search algorithms, on the other hand, are complete and optimal, provided that the heuristic part of the evaluation equation, $h(N)$, does not overestimate the distance to the target.
    - Un-informed search techniques usually use an explicit search tree as their State Space; on the other hand, informed search algorithms (like A* algorithm) have the ability to use graphs as their search space. The advantages of using graph search were explained earlier.

- Now, the reasons behind choosing the A* algorithm rather than other types of informed search algorithms are discussed:
    - The A* algorithm has an exact cost function $g(N)$ in addition to its heuristic function $h(N)$. Thus it is able to avoid the shortcomings of informed algorithms that depend only on heuristic functions which usually underestimate the path cost (Russell and Norvig, 2003).
    - The A* algorithm takes into consideration the distance already travelled by the resource. This is ideal to resource travelling problems since an educated estimate of a resource's total movement is needed, not only a segment of the movement.
    - Among all the algorithms that are considered optimal, there is no other algorithm that is guaranteed to expand to fewer nodes before reaching the shortest path than the A* algorithm (Russell and Norvig, 2003). This provides fast processing of shortest paths, sending timely feedback on resource movement planning to the simulation component of the framework.

[3] For further information on how these criteria are applied please see Russell and Norvig's *Artificial Intelligence: A Modern Approach* 2nd Edition, chapter 3.

- The A* algorithm is a typical single-pair shortest-path problem solving algorithm. The resources' shortest path problem falls in this category of problems. Each resource in a construction site has only a single origin point and a single destination that it moves between.

It is worth mentioning that since a graph (mesh grid) rather than a tree is being utilized here as the State Space, the A* algorithm is guaranteed to discover the shortest(optimal) paths onsite if, and only if, its heuristic function $h(N)$ is both admissible and monotonic (Russell and Norvig, 2003). To show that $h(N)$, which is equivalent to h_{sld}, would give the optimal problem solution, the following proof is demonstrated:

Figure 3-1 shows an extract of the mesh generated by the mesh generation mechanism to capture and model changes in a site's spaces. This extract is used to prove that the A* search algorithm will always give a resource's shortest path regardless of the current site layout.

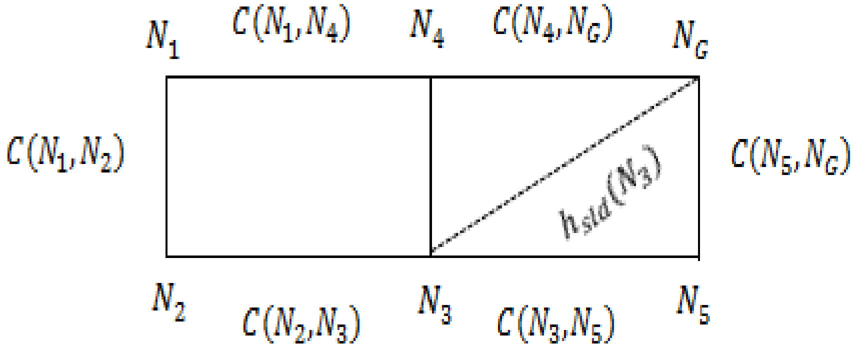


Figure 3-1 An extract of the node mesh (grid) generated by mesh generation mechanism covering a site to prove that the A* algorithm will give optimal results for this search problem

Since the A* algorithm is used here for a node mesh (Figure 3-1 and Figure 3-9), it will produce only optimum results (real shortest resource route) if and only if $h(N)$ satisfies the requirement of monotonicity (consistency) (Russell and Norvig, 2003).

The $h(N)$ function satisfies the requirement of monotonicity if for every node (N_i) and every successor node (N_s) of node (N_i), the estimated cost of reaching the goal node N_G from N_i is not greater than the step cost of getting to (N_s) from (N_i) plus the cost of reaching N_G from N_s.
That is $\forall n$, $h(N)$ is consistent if $\forall n$:

$$h(N_i) \leq c(N_i, N_s) + h(N_s) \text{ ……. } (2)$$

Therefore, to prove that the A* search algorithm, will give the shortest path for any mobile object in any given site layout, inequality (2) has to be proven true for the chosen mesh.

To prove that, Euclidian distances (straight line distance) are used inside the A* algorithm implemented in the HSV framework to estimate $h(N)$, $h(N)$ in this case becomes the $h_{sld}(N)$ which is equal to:

$$h_{sld}(N) = \sqrt{\left(N(x) - N_G(x)\right)^2 + \left(N(y) - N_G(y)\right)^2} \text{ …… } (3)$$

Where,

$N(x)$: X-Coordinate value of Node (N)

$N_G(x)$: X-Coordinate value of Goal Node N_G

$N(y)$: Y-Coordinate value of Node (N)

$N_G(y)$: Y-Coordinate value of Goal Node N_G

This makes inequality (2) a simple triangle inequality that will be satisfied for any node (N) on the mesh node covering the site. To demonstrate that in Figure 3-1 (above) the dotted line represents $h_{sld}(N_3)$, or the heuristic distance of node N_3 to the goal node. Further, according to inequality (2) the following should be true:

$h_{sld}(N_3), \leq c(N_3, N_5) + h(N_5)$

But, $h(N_5) = h_{sld}(N_5)$ since Euclidian distance is used inside the implemented A* algorithm; therefore, it can be represented as:
$h_{sld}(N_3) \leq c(N_3, N_5) + h_{sld}(N_5)$

Hence, from Figure 3-1, $h_{sld}(N_5) = C(N_5, N_G)$. Therefore,
$h_{sld}(N_3) \leq c(N_3, N_5) + C(N_5, N_G)$, which can be shown as true from Figure 3-1.

Therefore, $h(N)$ is Consistent $\forall n$, and the A* algorithm implemented inside the visualization component of this framework will produce the shortest paths for the various project resources given different site layouts.

It is worth mentioning that the A* search algorithm will give the shortest paths, but it will not search for the obstacle-free paths when the site layout changes. To overcome this, another mechanism was implemented to work hand-in-hand with the A* search algorithm — the mesh generation mechanism. This mechanism will change the generated node mesh every time a site's geometry is changed through an update to the site's spaces or layout by the project's simulation component, as it will be explained later in this chapter.

3.3 HSV Framework Development

As it was mentioned earlier, the representation of site's geometry is a prerequisite to the operation of the pathfinding mechanism. The site geometry changes are based on the project's progress. There is always a change in the site geometry due to: (1) Changes in site space which happen because of permanent structures monotonic addition to the site area, and (2) Changes in the locations and geometry of objects representing temporary site facilities. A project's progress is simulated through simulation based modeling; however this type of modeling does not have the ability to depict/model changes in site geometry. A visualization component that concurrently depicts the simulated project processes with corresponding changes in a site layout is added to this simulation modeling to model these changes in site geometry. That is, SDV is used here to enhance the site spatial data and geometry modeling.

In Chapter 2, some state of the art construction SDV mechanisms and frameworks that were explained in recent research were reviewed. Although these frameworks and mechanism have contributed towards SDV construction research, they have their limitations which were discussed in Chapter 2. These limitations prevent the effective utilization of these SDV frameworks in modeling the changes in site spaces that result from the simulation.

To effectively model the changes in site space that result from the simulation, this research proposes a loosely coupled simulation visualization framework. In this framework, the execution of the visualization components is parallel and independent from the simulation engine. Execution of these components can also be carried by completely separate and dedicated computer systems allowing the utilization of the strength of each component independently and exclusively. In addition, the framework capitalizes on existing high-level 3D graphic engines to reduce the effort required to customize and develop the visualization components.

This section discusses the proposed framework. The framework capitalizes on gaming technologies in the search for an effective method to visualize simulated construction operations and model both; the changes in site geometry and site space based on that. This section also presents the architecture of the framework and the incorporation of the 3D engines as the core of the visualization component of this framework. In addition, it discusses the benefits and challenges experienced during development.

The following sections will discuss further extension of the framework to utilize it in: (1) Solving the resources pathfinding problem, and (2) Planning onsite resources mobilization. This is based on the framework's ability to model changes in both site's geometry and site's spatial data.

Therefore, this initially develops the tool (framework) that:

- Depicts simulated construction processes, the final construction product development and the construction resources interactions on site together with their respective positions based on the project's simulation scenario.
- Depicts changes in site layout and site geometry based on the project's simulation. The current site geometry (spatial data and layout) are then used by the framework's PME to solve the pathfinding problem.

This section will explain the development of this new HSV framework.

3.3.1 HSV Framework and Underlying Distributed Simulation Standards

The proposed High Level Architecture (HLA)- based Simulation Visualization framework which can be referred to as HSV framework for short is based on the distributed simulation standard known as HLA (IEEE 1516.2000). Distributed simulation is a technology that enables models to be linked together over computer networks so that they work together (or interoperate) during a simulation run. HLA is a set of standards that regulates distributed simulation development (Taylor et al., 2003). In HLA, each simulation is connected with the others through Runtime Infrastructure (RTI) software. The simulations or models connected to the RTI are called Federates and a collection of them working together to simulate the same system is known as a Federation. A federate is not required to be a simulation model, but can be any piece of software or hardware, or even an interface to human user (user-in-the-loop), as long as each federate complies with

HLA standards (Rycerz et al., 2007). RTI facilitates communication between federates and performs time management of the simulation. A key component of a Federation (collection of federates) is the Federation Object Model (FOM) which defines all the entities produced by the various federates during the simulation run, states the attributes of those entities and information about which federates will update those entities' attributes (i.e. publish them), and which federates will read them (i.e., subscribe to them). For example, in the proposed HSV framework, federates responsible for the visualization may only subscribe to some attributes, while federates responsible for simulation behaviors generation can write and read (publish and subscribe) these attributes.

Figure 3-2 shows the HSV concept, typical components and communications between the visualization federate(s) on one hand and other functional/simulation federates on the other hand.

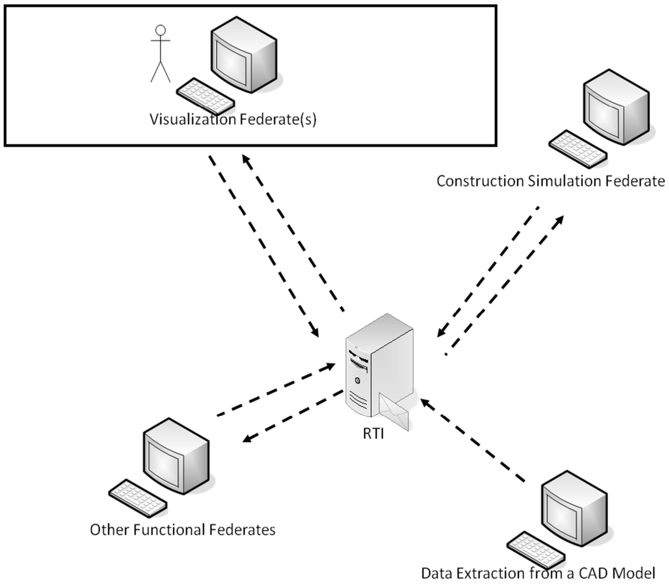

Visualization Federate(s)
Construction Simulation Federate
RTI
Other Functional Federates
Data Extraction from a CAD Model

Figure 3-2 HSV framework concept and communication

As shown in Figure 3-2, the following are typical components that are part of the framework or can be connected to it:

- Visualization Federate(s): Communicates with the source simulation or any other federate. It depicts the construction operation simulation concurrently as the simulation time advances to reflect what is going on in the simulation instantly. The visualization federate has a 3D engine such as Blender Game Engine (BGE) or TrueVision3D® at its core. Chapter 4 will showcase a comprehensive case study which involves the implementation of the HSV framework with a BGE based visualization component. Another smaller case study will have the visualization component of the HSV framework based on a TrueVision3D® engine will be discussed later in this chapter.

- Source Simulation Federate(s): That is where the simulation of the various construction processes takes place. Also, simulation time is usually advanced by these federates. They connect to their respective Viewer Federate through the RTI.
- Other Functional Federates: Any other federates that may provide supporting services to the federation. For example, a database federate or a CAD Model federate used to extract simulated objects' attributes values from external information systems or CAD systems to update them during the simulation run.

It is worth mentioning that the framework's development is a collective research team effort. The main components that were done through this research are the visualization federates, visualization engines and their connections, other developments such as the simulation components of the framework are beyond the scope of this research. All communications between the various federates go through the RTI using standard HLA protocols. Federates simulating various construction processes can be added or removed from the federation. Visualization Federate(s) can be connected to allow simultaneous visualization of various simulated construction operations. The visualization is concurrent with the simulation, which allows the user to see the simulation as it is happening and provides a better representation of the changes in a construction site's spatial aspects and the logic of the construction process.

The following section will discuss the development of the 2-way data communication between the simulation and visualization components of the framework.

3.3.2 The Two-Way Communication between the Simulation and Visualization Components and Framework Time Management

The communications between the simulation and visualization components of the framework are done through several layers. The visualization federates reflect the simulation's object classes attributes values as they are updated by the simulation models. These reflections are communicated between the two federates through the RTI of the HLA configuration. On the other hand, the viewer federates and the 3D visualization engine(s) communicate through a network User Datagram Protocol (UDP), which provides language-independent communications between the .NET based HLA federation and 3D engine.

This UDP connection allows communication in both ways, where each of the Viewer Federates and the 3D engine act as both servers and clients. The Visualization Federate does not act only as a client to send simulation messages to the 3D engine, but also acts simultaneously as a server to receive information from the 3D engine. The same goes for the 3D engine, which acts as a client to send information to the running simulation, and at the same times acts as a server to receive the changing values of the simulation object class attributes. This 2-way communication development was necessary in order for the 3D engine to depict and model an updated site layout (geometry), which changes as the simulation is running (dynamic site layout), and in turn send the site pathfinding analysis results to the running simulation as it will be described in the next case study chapter. Figure 3-3 below explains the simple concept behind this two-way communication Network UDP connection used in the HSV framework.

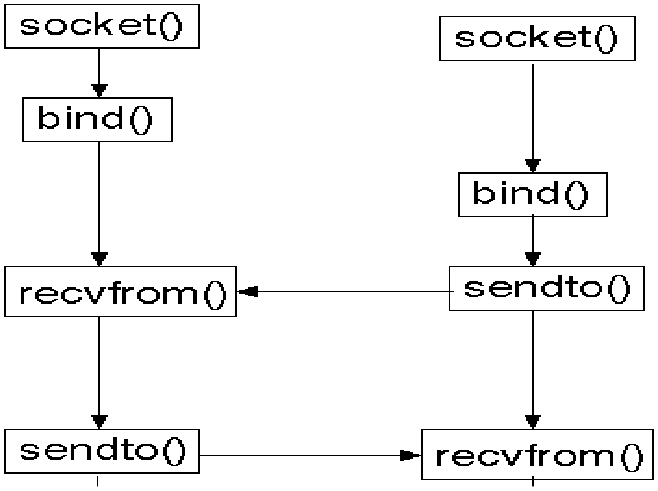


Figure 3-3 The fundamental concept behind the Network UDP

Different ports have to be assigned for sending and receiving per application. The use of a network UDP method as a means of two-way communication has built on the strength already inherent in distributed simulation HLA standards, by allowing the site visualization 3D Engine component and the construction simulation component to physically work—exist on—different terminals connected through networks. This is an asset in itself, as it allows different members of a construction team to assess the outcomes of running the framework from different physical locations, evaluating different simulation scenarios. Operating the framework using federates which are distributed to different physical locations was already tested and proven successful. In this case, the connection takes place on different ports with different IP addresses instead of having all ports on the local host. The sending and receiving ports on each side are conceptually explained in Figure 3-4 below.

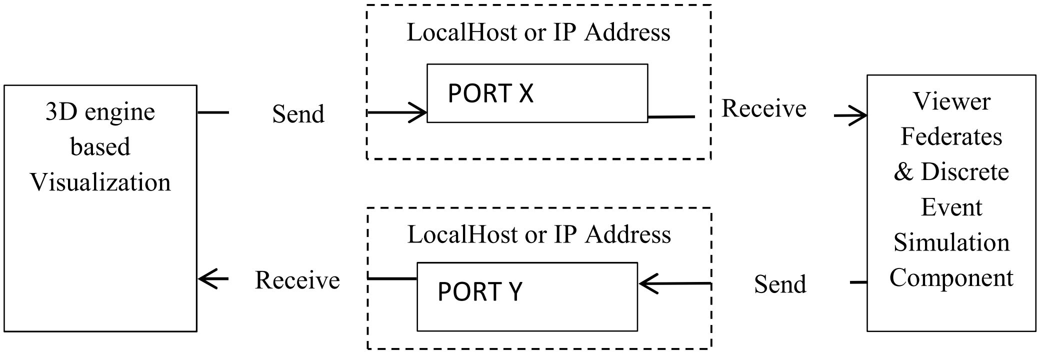

LocalHost or IP Address
PORT X
3D engine based Visualization
Send
Receive
Viewer Federates & Discrete Event Simulation Component
LocalHost or IP Address
PORT Y
Receive
Send

Figure 3-4 Different ports could be on different workstations or on the same local host

It was necessary to make the visualization component talk back to the simulation component in order to send the results of running the pathfinding mechanism on the current site's layout geometry back to the construction simulation during its run as it will be shown in the next Chapter.

Another interesting aspect of this framework is how the simulation time is synchronized between the simulation and visualization components of the framework. Time management inside the framework is organized through strict HLA IEEE standards. These standards set certain federates to be time-regulating and others to be time-constrained. This means that the overall simulation time of the federation is controlled through time regulating federates which do not allow the global simulation time of the federation to advance unless they send a "Time Advancement Grant" to the RTI. The grant is sent once all the simulation events that should be processed before the requested advancement time (time the federation will advance to) are fired.

On the other hand, time-constrained federates, such as the Visualization Federate(s); advance their times according to the messages from the Time Regulating Federates (Simulation Federates).

The next chapter (Chapter 4) will showcase a comprehensive case study that implements the integrated HSV framework with its PME and applies them to a real construction problem putting both; the framework's ability to model sites' spatial data, and its PME's ability to solve a complicated construction pathfinding problem to use. However, it was necessary, before implementing the pathfinding mechanism into the framework, to make sure that the suggested framework architecture will allow the various simulation and visualization components to interoperate and function in the desired way. In order to do that, a testing development in which various simulation and visualization federates are connected through the HLA standards and UDPs to create a construction bidding simulation game was done. This bidding simulation game development is explained in Appendix I.

3.4 Implementation and Operation of Pathfinding Mechanism Extension (PME)

In this section the overall pathfinding mechanism consisting of the mesh generation mechanism and the A* algorithm will be detailed. The pathfinding mechanism is the core of the PME that will be implemented inside the 3D engine of the HSV's visualization component. The next chapter will showcase a comprehensive example that explains the pathfinding process and how the construction industry benefits from it.

A set of flow charts below, Figures 3-5 to 3-8, explain the pathfinding mechanism implemented inside the HSV framework's visualization component 3D engine.

Figure 3-5 conceptually explains the data exchange between the simulation and visualization components during the simulation run, together with the inputs and outputs to the pathfinding mechanism.

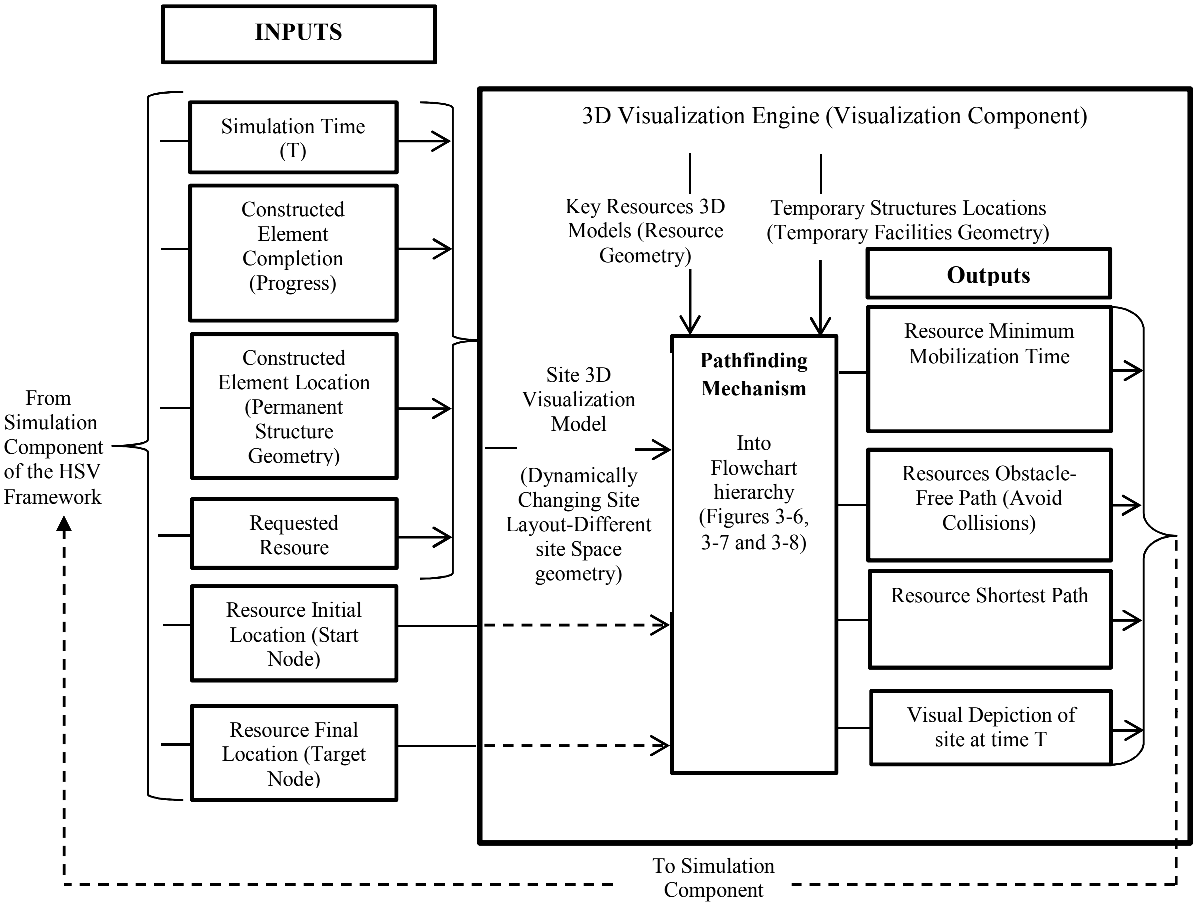

INPUTS
Simulation Time (T)
Constructed Element Completion (Progress)
Constructed Element Location (Permanent Structure Geometry)
Requested Resoure
Resource Initial Location (Start Node)
Resource Final Location (Target Node)
From Simulation Component of the HSV Framework
3D Visualization Engine (Visualization Component)
Key Resources 3D Models (Resource Geometry)
Temporary Structures Locations (Temporary Facilities Geometry)
Site 3D Visualization Model
(Dynamically Changing Site Layout-Different site Space geometry)
Pathfinding Mechanism
Into Flowchart hierarchy (Figures 3-6, 3-7 and 3-8)
Outputs
Resource Minimum Mobilization Time
Resources Obstacle-Free Path (Avoid Collisions)
Resource Shortest Path
Visual Depiction of site at time T
To Simulation Component

Figure 3-5 Conceptual data exchange between simulation and visualization components of the framework and inputs and outputs to the pathfinding mechanism

Figure 3-6 explains the overall operation of the pathfinding mechanism. It explains the interoperability between the mesh generation mechanism and the A* algorithm. Therefore, this is referred to as the pathfinding mechanism Parent Flowchart since it is at the top of the hierarchy. This Parent Flowchart provides a graphical representation for the overall process explained in this section.

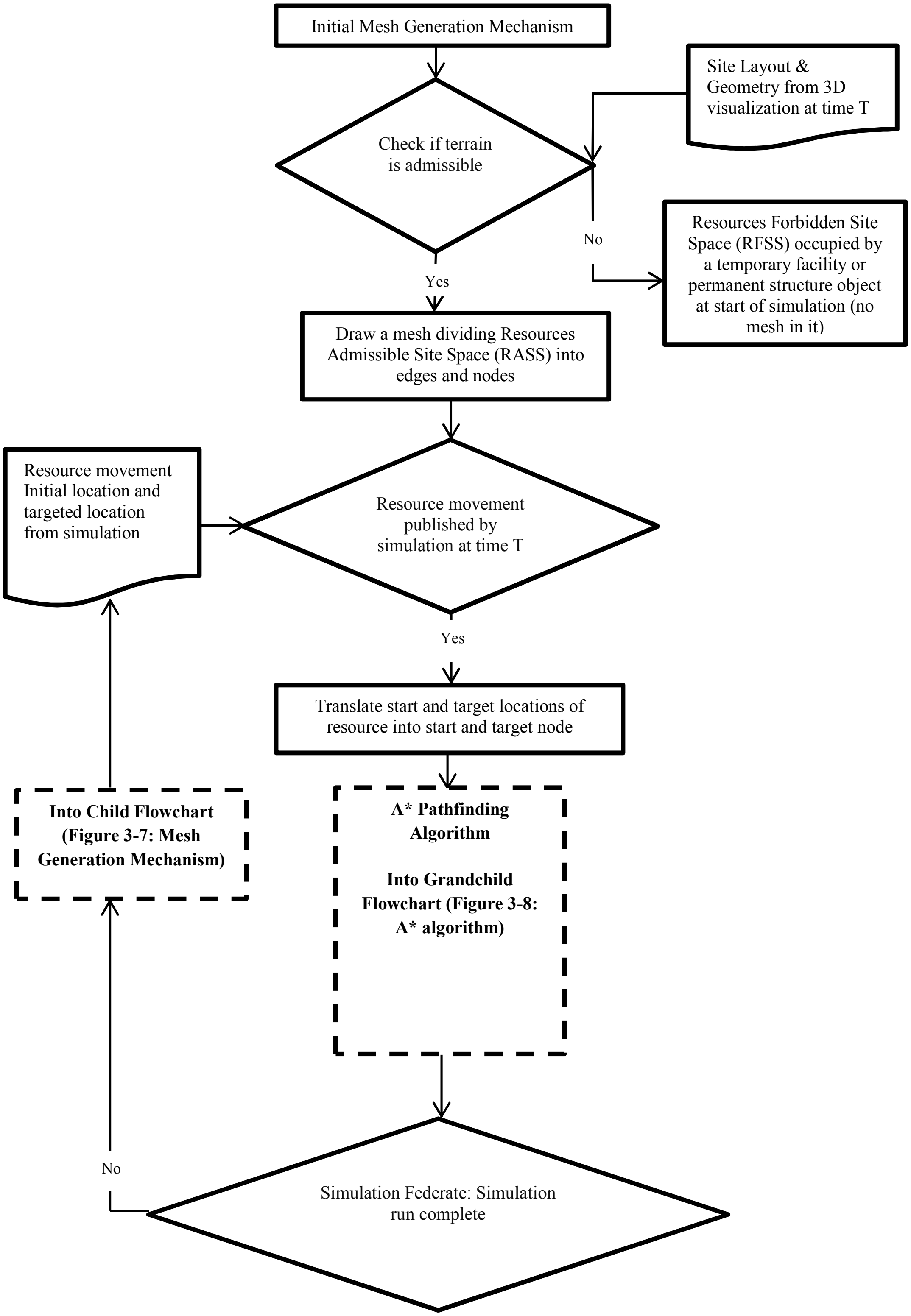

Initial Mesh Generation Mechanism
Check if terrain is admissible
Site Layout & Geometry from 3D visualization at time T
No
Resources Forbidden Site Space (RFSS) occupied by a temporary facility or permanent structure object at start of simulation (no mesh in it)
Yes
Draw a mesh dividing Resources Admissible Site Space (RASS) into edges and nodes
Resource movement Initial location and targeted location from simulation
Resource movement published by simulation at time T
Yes
Translate start and target locations of resource into start and target node
Into Child Flowchart (Figure 3-7: Mesh Generation Mechanism)
A* Pathfinding Algorithm
Into Grandchild Flowchart (Figure 3-8: A* algorithm)
No
Simulation Federate: Simulation run complete

Figure 3-6 HSV pathfinding process Parent Flowchart showing interoperability between the mesh generation mechanism and the A* algorithm

Figure 3-7 explains the mesh generation part of the pathfinding mechanism. It is the Child Flowchart since the mesh generated by the mesh generation mechanism is a prerequisite to the operability of the A* algorithm. This Child Flowchart provides a graphical representation for the process explained in Section 3.4.1 of this chapter.

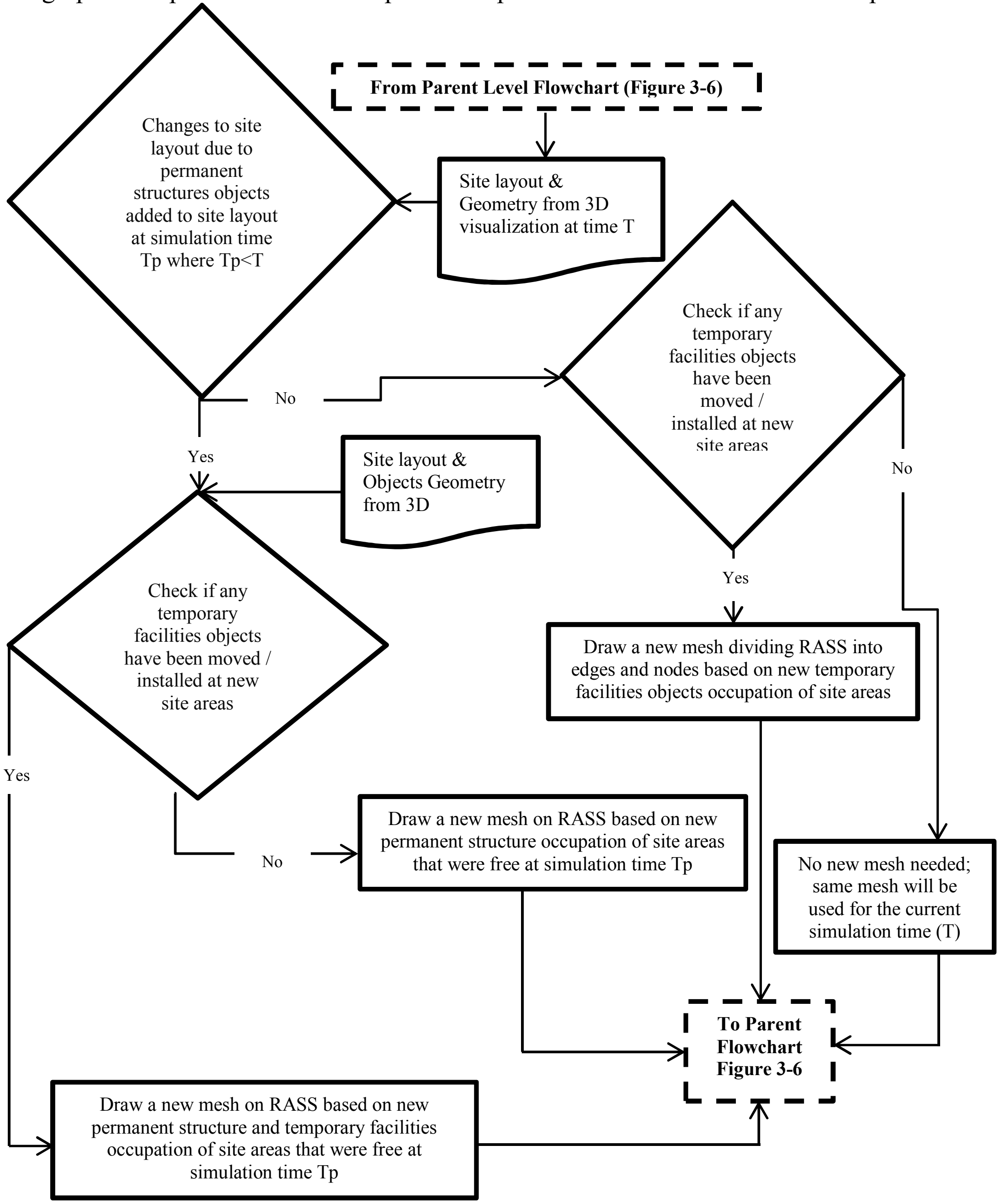

From Parent Level Flowchart (Figure 3-6)
Site layout & Geometry from 3D visualization at time T
Changes to site layout due to permanent structures objects added to site layout at simulation time Tp where Tp<T
No
Yes
Check if any temporary facilities objects have been moved / installed at new site areas
Site layout & Objects Geometry from 3D
Check if any temporary facilities objects have been moved / installed at new site areas
Yes
No
Yes
No
Draw a new mesh dividing RASS into edges and nodes based on new temporary facilities objects occupation of site areas
Draw a new mesh on RASS based on new permanent structure occupation of site areas that were free at simulation time Tp
No new mesh needed; same mesh will be used for the current simulation time (T)
To Parent Flowchart Figure 3-6
Draw a new mesh on RASS based on new permanent structure and temporary facilities occupation of site areas that were free at simulation time Tp

Figure 3-7 Child Flowchart explaining the concept of the mesh generation mechanism

Figure 3-8 explains the A* algorithm part of the pathfinding mechanism. It is the Grandchild Flowchart since its operation depends on the mesh generation mechanism dividing the Resources Admissible Site Space (RASS) into nodes and edges. This Grandchild Flowchart provides a graphical representation for the process explained in Section 3.4.2 of this chapter.

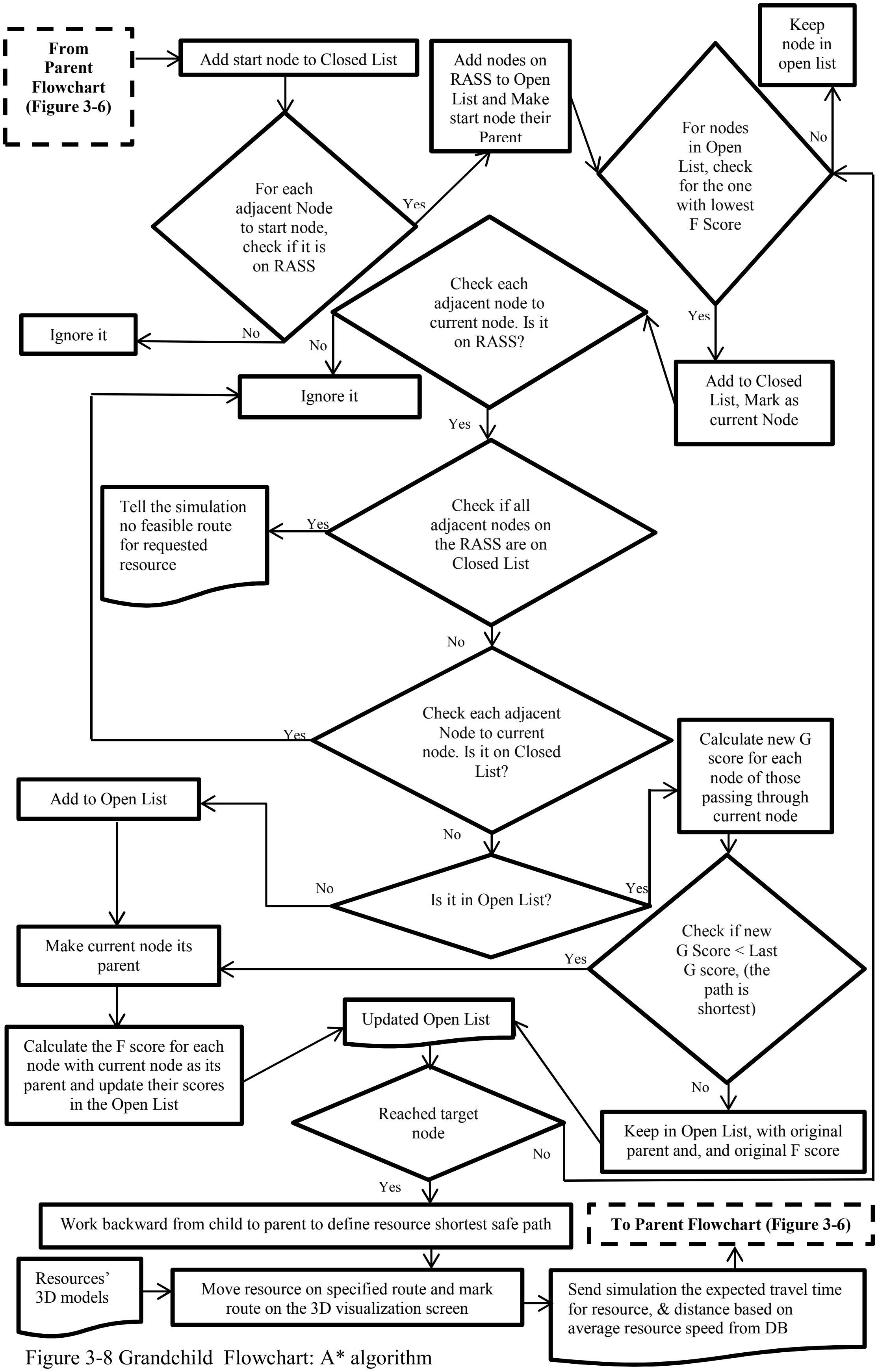

From Parent Flowchart (Figure 3-6)
Add start node to Closed List
For each adjacent Node to start node, check if it is on RASS
Yes
No
Ignore it
Add nodes on RASS to Open List and Make start node their Parent
For nodes in Open List, check for the one with lowest F Score
No
Keep node in open list
Yes
Add to Closed List, Mark as current Node
Check each adjacent node to current node. Is it on RASS?
No
Ignore it
Yes
Check if all adjacent nodes on the RASS are on Closed List
Yes
Tell the simulation no feasible route for requested resource
No
Check each adjacent Node to current node. Is it on Closed List?
Yes
No
Is it in Open List?
No
Yes
Add to Open List
Calculate new G score for each node of those passing through current node
Check if new G Score < Last G score, (the path is shortest)
Yes
No
Make current node its parent
Calculate the F score for each node with current node as its parent and update their scores in the Open List
Updated Open List
Keep in Open List, with original parent and, and original F score
Reached target node
No
Yes
Work backward from child to parent to define resource shortest safe path
To Parent Flowchart (Figure 3-6)
Resources' 3D models
Move resource on specified route and mark route on the 3D visualization screen
Send simulation the expected travel time for resource, & distance based on average resource speed from DB

Figure 3-8 Grandchild Flowchart: A* algorithm

The overall pathfinding mechanism will be explained first in this section with its inputs and outputs from and to simulation components respectively. After that, the two main components (mesh generation mechanism and A* algorithm) integrated to make up the mechanism will be explained in separate sub-sections.

Before the simulation run starts, a complete 3D prototype of the construction site is produced inside the visualization component's 3D engine. This prototype, which can be imported from AutoCAD 3D or other, drafting packages, contains the initial layout and geometry of the construction site at time zero (i.e., before the mobilization phase of the key mobile resources commences or any objects representing permanent structures geometry added). Also, a repository containing CAD models representing; key mobile resources, products and permanent structures should exist for the simulation visualization of the construction processes. The key mobile resources are those characterized by the following:

- Used to perform critical project activities
- Expensive to mobilize in terms of time and rental costs.

A construction decision-maker plans ahead for the utilization of resources. Also, these are the resources that are affected by the dynamic changes in site layout that result from changes in site objects (temporary and permanent facilities representations) geometry.

In this initial site prototype, the decision-maker starts to assign locations to the 3D objects representing temporary site facilities such as engineering offices or rebar workshops. These 3D objects' locations and geometry can change throughout the course of the project, although these changes are not part of the inputs that usually come from the project's Construction Simulation Federate. The HSV framework gives the decision-maker the ability to continuously update temporary site facilities representations through the visualization screen during the course of the simulation run.

Once the initial site plot at construction time zero is produced and temporary site facilities 3D object representations' locations and geometry are represented, the visualization component depicts the initial site layout. Once the simulation starts running, the visualization component will start to depict the various construction processes as well as the changes in site space that happen as a result of these processes and the work progress.

The visualization component's 3D engine depicts the changes in site layout and site geometry by receiving updates on the following from the Construction Simulation Federate(s) (inputs shown in Figure 3-5):

- Changes in Product Class attributes (permanent structural elements):
    - Updates on the product progress.
    - Updates on the product's location and geometry onsite.
- Changes in Mobile Resources Class attributes:
    - Resource to be utilized during certain periods of the project.
    - Resources' site locations during a certain period of the project.
- Interactions between the object classes mentioned above:
    - Interactions of elements of the final product with mobile resources during the construction process. For example, a module being in lift by a crane before getting it to its final set point.
- The simulation time.

The visualization component receives these messages concurrently as the simulation is running. The pathfinding mechanism then uses these inputs from the simulation component to find the least-expensive, safe path for each resource mobilization event and sends a feedback analysis of the mobilization event to simulation. The outputs are shown in Figure 3-5.

The operation of the pathfinding mechanism inside the Visualization Federate can be divided into two major phases:

- The mesh generation mechanism phase: The mesh generation mechanism categorizes the modeled site spaces into: Resources Admissible Site Space (RASS), and Resources Forbidden Site Space (RFSS). The RASS is the site space that is not occupied by objects representing temporary facilities or permanent structures and can be used by mobile objects (depending on these objects geometry and the space geometry) for manoeuvring and/or form a part of the object's road geometry. On the other hand RFSS is the site space that is occupied by objects representing temporary facilities or permanent structures and cannot be used by mobile objects for manoeuvring and does not form a part of the object's road geometry. Section 3.4.1 explains the operation of the mesh generation mechanism. The way the mechanism uses the HSV framework's ability to model site spaces' geometry changes occurring during the project's simulation to continually changes spaces from RASS to RFSS or vice versa is also explained. The mechanism's operation is illustrated in Figure 3-7.

- The A* algorithm phase: The algorithm uses the mesh generation mechanism's output. Section 3.4.2 explains the operation of the algorithm. The algorithm's operation is illustrated in Figure 3-8.

Both the mesh generation mechanism and the A* algorithm are part of the PME to be implemented inside the 3D engine of the visualization component, Figure 3-6 illustrates the interoperability between both components to solve the pathfinding problem in a continually changing site geometry.

3.4.1 Phase 1: Mesh Generation Mechanism[4]

The mesh generation mechanism is the basis of the application of the A* algorithm. The A* algorithm makes use of the nodes and edges to mark the RASS that will be processed by the algorithm to find the least-expensive, safe path for a certain mobile resource 3D object representation. Figure 3-9 shows the mesh covering a site layout's RASS. This same site will be used in next chapter as a case study for the HSV framework implementation.

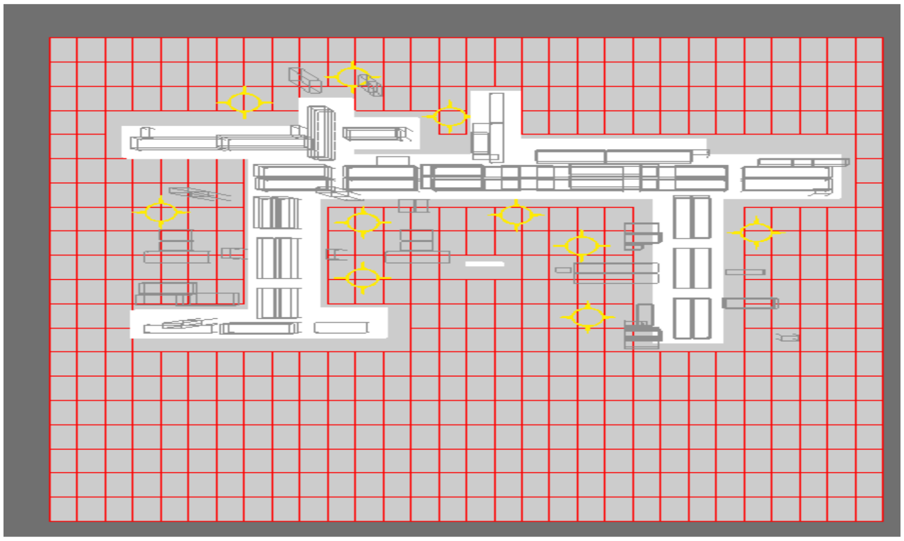


Figure 3-9 Mesh covering the construction site's RASS at simulation time zero

[4] Appendix IV contains the code for implementing the Mesh Generation Mechanism in the HSV framework's visualization component

The mesh covers the space of the site that can be utilized for mobile resources mobilization. Hence in Figure 3-9, the areas that are occupied by the footings and other representations of temporary structures are not covered by the grid. As a construction site dynamically evolves throughout the various stages of the project, and the site space geometry changes, the mesh continuously changes to accommodate these site changes. In other words, the site's space changes state between admissible and forbidden (RASS and RFSS).

The initial mesh generation takes place at simulation time zero based on the initial site layout. The programming code to produce the mesh is implemented inside the 3D component and is triggered once the visualization starts.

The mesh updates the available site spaces' categorization (either RASS or RFSS) once the project simulation component updates the project progress by adding permanent structural elements to the site plot which in turn adds 3D object representations of these elements to the site layout on the visualization screen. Also, the mesh is updated automatically when the decision-maker changes/experiments with various temporary facilities 3D object representations' geometry or locations during the simulation run.

Next, the application of the mesh generation algorithm inside the HSV framework's visualization component will be explored. The algorithm for mesh generation can be utilized for covering 2D and 3D spaces. Managing tempo-spatial aspects of resource mobilization and deciding on shortest paths geometry is only a 2D problem. Therefore, the site's spatial data are defined by only the x and y axes. Then the site layout and its limits are defined in terms of the global coordinates (on the x and y axes). The center coordinates are calculated and the user inputs the number of nodes along each axis. The more nodes covering the site space in each direction, the more complicated the A* algorithm calculations will be with each resource movement. The external borders of the site space are then defined for the x and y coordinates as follows:

$$X_{min} = X_c - X_l\,/2 \ldots\ldots (4)$$

$$X_{max} = X_c + X_l\,/2 \ldots\ldots (5)$$

$$Y_{min} = Y_c - Y_l\,/2 \ldots\ldots.\ (6)$$

$$Y_{max} = Y_c + Y_l\,/2 \ldots\ldots. (7)$$

Where X_{min} and Y_{min} are the minimum coordinates defining the borders of the site on one side along the x and y axes, respectively. X_{max} and Y_{max} are coordinates defining the borders of the site lot from the other side along the x and y axes, respectively. X_c and Y_c define the center coordinates parallel to the x and y axes, respectively. X_l and Y_l represent the site plot dimensions along the x and y axes, respectively.

Once the site lot is defined and the number of nodes along each axis is defined, it is time to create the initial mesh. First and for the sake of constructing the mesh, the algorithm gets the coordinates of each node forming the mesh covering the site, so each node becomes an [x,y,z] space coordinate node. Also, each node's adjacent nodes (to the north, south, east and west) are defined and stored as coordinates. Then the site spaces that are occupied by temporary site facilities are subtracted from the initial mesh. The mesh generation mechanism does this through the mechanism's code.

When the mesh generation code is processing a certain node, it checks if there are any objects representing permanent structures or temporary facilities on this node itself or in the distance between this node and its neighbor nodes. If an object exists, then this space is labeled as RFSS. This means that this space will not be covered by the mesh and subsequently will not be considered by the A* pathfinding algorithm when trying to find the shortest resources mobilization path on this site layout.

Now the mesh is ready to generate in the form of nodes and neighboring nodes' coordinates that cover only the RASS. This mesh code stores the mesh data in a global variable so that other modules and functions inside the HSV framework can access it during the simulation run. This covers the basics of the initial mesh generation shown in the Parent Flowchart (Figure 3-6).

In the following paragraphs, the way the node mesh changes during the simulation run itself, to accommodate dynamic changes in site layout during a simulation run, will be explained (Figure 3-7 Child Flowchart).

First, for permanent structural elements, the simulation time advances from simulation time (Tp) to current simulation time (T) on the DES timeline (where Time Tp precedes Time T). This means that the project has progressed. Consequently, new structural elements might now be occupying site spaces that were marked as RASS before reaching simulation time Tp. This means that those spaces now (at current simulation time T) should be marked as RFSS. Once a 3D object is added to the site spaces modeled by the HSV framework on the visualization screen, mesh is automatically regenerated to accommodate the changes in the site space geometry as a result of the permanent structures objects representations added to these spaces.

Unlike permanent structural elements, temporary facilities are not accounted for during the simulation. In other words, mobilizing or demobilizing a temporary facility is not an event that is triggered by the Construction Simulation Federate(s). In real construction sites, however, changes to temporary site facilities might occur due to certain issued change orders or unforeseen project conditions. On the visualization screen, certain 3D block representations with changing dimensions are first added to the site layout. These blocks—representing temporary construction facilities—can be manually controlled in terms of their dimensions and their site locations by the decision-maker throughout the project's simulation lifecycle. Once the block representation(s) of a temporary facility is moved or has its geometry changed, this automates a mesh re-generation code that regnerates the node mesh to accommodate the changes in the site space geometry as a result of the temporary facilities added to the site spaces.

The way the mesh generation codes and modules operate will be further clarified in the next case study chapter.

3.4.2 Phase 2: A* Algorithm Application[5]

Once the site layout is covered by an imaginary mesh of nodes and edges after Phase 1, the A* algorithm can be applied to plan and analyze the triggered simulation's resource mobilization events.

The nodes (vertices) of this mesh represent the travelling points, while the edges represent the feasible paths between these nodes. The A* algorithm goes through these nodes and edges to find the shortest path and yield the distance between a resource's original and destination points as part of the resource's road geometry details when the simulation schedules a resource mobilization event. In this section, the way the A* algorithm operates inside the visualization component of the HSV will be explained (Grandchild Flowchart, Figure 3-8).

[5] Appendix V contains the code for implementing the A* Search Algorithm in the HSV framework's visualization component

The A* algorithm now has all the inputs it needs: the current site layout mesh (from mesh generation mechanism output) and the various site elements 3D models (from the 3D engine), along with a resource's start and target nodes or locations (from simulation federate). The way the algorithm finds the obstacle-free, shortest path is through the following typical steps:

1) The algorithm adds the resource's start node to a list of nodes called Closed List.

2) It then processes the start node's neighboring nodes as follows:

a) Look for nodes adjacent to the start node and add them to a node list called Open List. Make the start node their parent. In the Open List look for the node that has the lowest travel cost based on the $f(N)$ evaluation function.

b) Move the node with the lowest $f(N)$ to the Closed List and call it the current node. Leave the other adjacent nodes in the Open List.

3) The following steps are repeated until the target node (resource's target location on site) is reached:

a) For each of the nodes adjacent to the current node:

- If the node is in the RFSS or if it is in the Closed List, it is ignored. Otherwise:
  - o If the adjacent node is not on the Open List, it is added to the Open List. The current node is made the parent of this node. The $f(N), g(N)$ and $h(N)$costs of the added node are recorded.
  - o If the adjacent node is already on the Open List, the algorithm checks to see if the path to that node going through the current node is better, using $g(N)$ cost as the measure. A lower $g(N)$ cost means that this is a better path. If the $g(N)$ is lower going through the current node, the algorithm changes the parent of the node to the current node, and recalculates the $g(N)$ and $f(N)$ scores of this child node.

b) The A* algorithm stops when:

- The resource's target node is reached and added to the Closed List, in which case the path has been found.
- The algorithm has gone through the maximum number of nodes defined by the user of the framework or the Open List becomes empty after going through all the nodes on the current site layout mesh, yet failing to reach the target node. This means that there is no feasible route for the requested resource mobilization given the current site layout. In this case the decision-maker has to remove the obstacles from the resource's way or find another means of performing the wanted construction operation.

4) Saves the chosen path: If a path is found, working backwards from the target node, the algorithm goes from each node to its parent node until the starting node is reached. Once the route is known it is marked on the site layout and all the route information is sent back to the simulation.

3.4.3 PME Limitations and Assumptions

The Mesh Generation Mechanism and A* algorithm implemented in this framework have their limitations when it comes to searching for the mobile resources shortest paths. The following limitations can be summarized as follows:

- The mesh generation mechanism does not take in consideration the change in terrain that result from changes in site topography during its operation. A terrain with steep slope can hinder the mobilization of heavy lift mobile resources and should be represented as a RFSS. The PME implemented here does not consider such a space to be a RFSS.
- The mesh generation mechanism leaves a safety buffer zone marked as RFSS around permanent structures to assure that the shortest path chosen by the A* algorithm would not cause any collisions for the resource when mobilized on it. This method assumes that the travelling mobile resources have a uniform shaped print (for example square or rectangular print). This method does not allow full collision detection when it comes to resources with irregular shaped prints.

3.5 The HSV Framework Implementation

This chapter introduced the HSV framework and focused on its role in modeling changes in site space that could not be modeled by typical simulation modeling. Once the interoperability between simulation and visualization components of the framework was proved to be functioning, a further step was taken to build on this framework's ability to model changes in both: site spaces, and layout that occur as a result of the construction processes that take place in a project. This was through a PME that is implemented in the visualization component of this HSV framework. This PME has a pathfinding mechanism that uses the aforementioned ability of the framework to find the resources' shortest safe paths on site. This is an application of a typical intelligent agent's pathfinding problem in the construction research.

To demonstrate how the HSV framework and its PME interoperate, a comprehensive case study that implements the HSV framework with its PME and applies them to a real construction problem will be demonstrated in the next chapter. The case study puts both; the framework's ability to model sites' spatial data based on construction operations simulation, and its PME's ability to solve a complicated construction pathfinding problem to use. The case study shows how the framework was used to solve a heavy lift mobile resources tempo-spatial management problem in a real construction site.

Chapter 4: Case Study: Using HSV Framework and its PME for Tempo-Spatial Planning of Heavy Lift Cranes

4.1 Introduction and Case Study Explanation

To explain and test the developed framework, it was applied to the spatial management of heavy lift cranes at a real construction site. A real industrial construction project (Constructing Crude Oil Upgrader in Alberta) was used for implementing the integrated HSV framework with its Pathfinding Mechanism Extension (PME) to demonstrate how the framework can contribute to planning heavy lift resources mobilizations in a dynamically changing site layout. This case study will show how the HSV framework and its PME solve the heavy lift cranes tempo-spatial management problem in a real construction site.

4.1.1 Case Study Definition and Project Specifics

In order to model the changes in site spaces and site layout that occur as the project progresses, and to plan heavy lift cranes' mobilizations in the oil upgrader project site through the PME, the HSV framework firstly has to provide a 3D visual representation of simulation behaviors generated by an industrial construction federation. The industrial construction federation models industrial construction operations in general and was used to model the oil upgrader project. The site involved in this case study is shown in Figure 4-1. The project represents the construction of an upgrader for crude oil. It is built using preassembled pipe spool modules. Pipe spools are manufactured off-site in a fabrication shop and completed spools are then passed to a module assembly yard. In the module yard, each module is allocated a space (bay) for completing its assembly. The pipe spools that comprise a module are moved to the bay where the assembly will take place. Once modules are assembled, they are shipped to the site and placed into position using mobile cranes. The site construction sequence depends on the module installation schedule and whether its predecessor has already been installed in its final location on site.

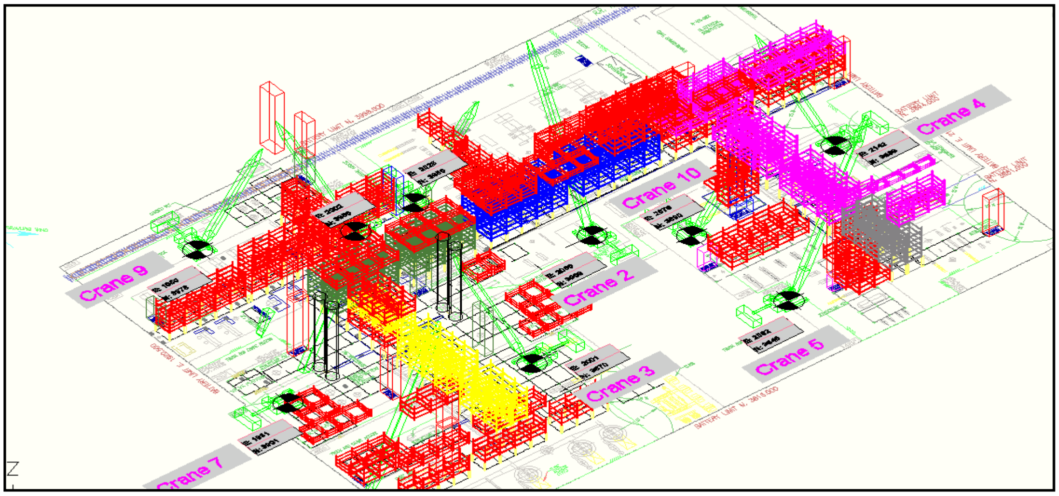


Figure 4-1 PCL-Shell Ft. Saskatchewan site

4.1.2 Case Study Problem Definition and Solving

The contractor wanted to find a way to study/model future changes in both: site's space availability and layout that happen due to changes in permanent structures and temporary facilities as the project progresses. Also, the contractor wanted to study the effects of these changes in site layout/spaces and their effect on the heavy lift cranes mobilization activities durations. The scheduled heavy lift crane mobilization events take place when a heavy lift crane is scheduled to perform a module lift at a certain location on site. Each lift involves a heavy-lift crane mobilization process preceding it. Four different heavy-lift mobile cranes (two 880-ton, one 660-ton, and one 400-ton) were used to lift the modules to their final set points. There are more than 50 heavy lifts to take place which requires more than 50 crane mobilization events throughout the project. These heavy lift crane mobilization events are expensive and have to be accounted for in the overall project schedule towards the overall project duration. It is a difficult task to ascertain the shortest and at the same time obstacle-free mobilization route for each of these cranes in a changing dynamic site layout. It is also hard to estimate the mobilization duration required for each of the module's lifting activity. The contractor wants to make sure that the duration and costs of heavy lift cranes mobilization events are minimized through finding the shortest safe crane paths (in terms of activity cost and activity duration) in the continually changing site layout (geometry).

A planning tool was needed to predict these changes in site space and to choose and evaluate the various mobilization routes for each of the cranes. This tool can help in decreasing the overall project duration and save money in terms of the cranes' rental costs, eventually leading to savings in the total project costs. This case study chapter describes how the framework described in Chapter 3 solved this heavy lift cranes tempo-spatial management problem and act as an accurate estimation tool for the resource mobilization durations.

The specific location of any of the heavy-lift cranes or modules onsite is depicted through the Visualization Federate of the framework. The lifting event involves the mobilization of a crane from an initial to final point. The lift locations are predefined onsite for each set of modules. Figure 4-2 shows the site layout on the visualization screen, where yellow ring markers show these predefined potential lift locations. Once the Discrete Event Simulation (DES) component of the framework triggers a lifting event and chooses a crane, the crane moves on the visualization screen from its initial location to its lifting location (final location), where it will install the modules. The pathfinding mechanism will find the shortest, safe path for each crane and the expected duration of this mobilization. The framework's PME will look for the shortest paths in a site prototype created by the simulation visualization. This prototype has changing layout that mimics the expected changes in the real site layout as the project progresses. The path for each lifting process will vary depending on the current site layout during the project's lifecycle.

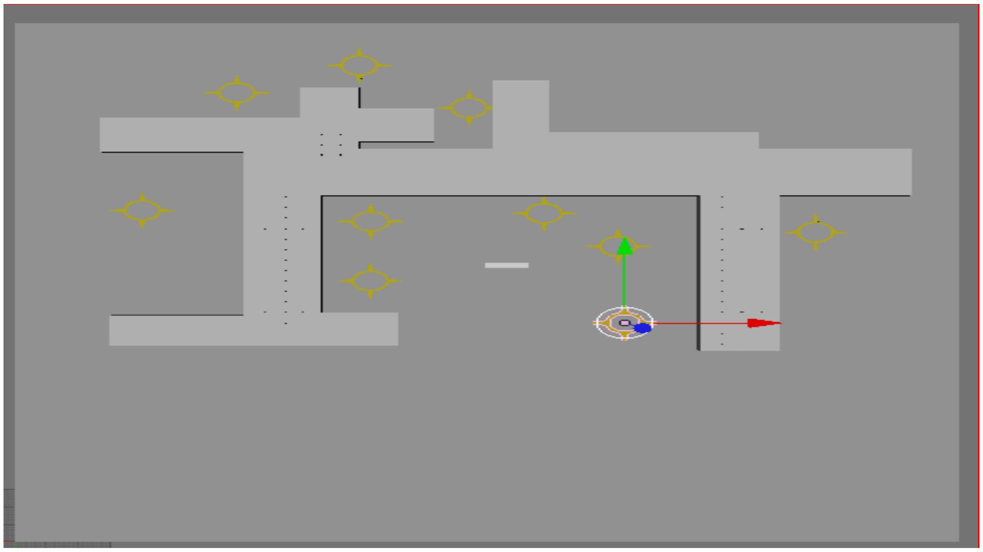


Figure 4-2 Screen from Visualization screen Blender Game Engine (BGE) showing the 10 yellow marker 3D objects marking the 10 potential lifting locations inside the site layout plan

4.1.3 Case Study Chapter Flow and Problem Solving Mechanism

To solve this problem, the HSV framework with its PME was applied to this site through the steps described in Chapter 3. The solution presented in this chapter follows the steps taken to solve the problem. Firstly, the generic architecture of the HSV framework presented in Chapter 3 was modified to be implemented in this problem. The framework's architecture is presented. The framework then needs to visualize the simulated construction processes. To do that the various simulation object classes attributes changes are transformed from their respective simulation states to visual behaviors in the Blender Game Engine (BGE) of the framework's visualization component. The ways the construction elements' various simulation states are mapped into visual behaviors using the visualization component are explained. Also, the ways the heavy lift cranes' various simulation states are mapped into visual behaviors will be explained. The chapter then describes the way the various simulated construction processes that cause changes in site space geometry and/or site layout are visualized. Next, the way the BGE based visualization models changes in site's spaces that happen as a result of these simulated construction processes is described. This modeling of site space depicts the dynamic changes in both: site layout and site geometry. These change in site's layout and geometry act as an input to the framework's pathfinding mechanism. The way the visualization component's pathfinding mechanism uses these modeled site spaces to find the heavy lift cranes shortest safe paths is then explained.

Finally, screen shots and numerical demonstrations which showcase how the pathfinding mechanism finds paths for various crane mobilization events triggered by the simulation is shown. These screen shots and numerical demonstration will show how the pathfinding mechanism explained earlier in Chapter 3 is applied to find different shortest paths for various crane mobilization events. To demonstrate the pathfinding mechanism abilities, it will be used to give numerical evaluation of different crane routes defined for the same crane mobilization event given different site layouts resulting from geometrical changes in any of the site's permanent structures or temporary facilities. The way the pathfinding mechanism sends numerical assessments of each resource's mobilization shortest feasible route to the simulation component during the simulation run is also explained. All of these visualizations and pathfinding processes are carried concurrently during the simulation run as it will be shown. Also, the way that the decision maker can try various layout scenarios during the simulation run is explained.

4.2 Site Space Modeling and Heavy Lift Cranes Mobilization Planning in Industrial Construction Simulation Case Using HSV Framework and its PME

A federation was built based on the architecture explained earlier in Chapter 3 to model the whole operation from spool fabrication to site construction. The visualization of the following simulated construction operations takes place:

- Assembly of modules at assigned bays: Each module is built out of number of pipe spools at its assigned bay. The HSV framework depicts the processes of modules' assignment to bays in addition to the building of modules inside bays. A specific module's assignment and its progress at any time during the simulation run (project duration) is shown through the Visualization Federate. The way that this depiction is achieved will be explained further in the following sections.
- Modules installation on construction site: This involves the use of cranes to install modules at their final location based on predefined construction logic as will be explained below. The specific location of any of the heavy lift cranes or modules on site is shown through the Visualization Federate. The corresponding lift location of a module and its placement time (to its final set point) can be known through the visualization. The mechanism behind this depiction is explained in the following sections. The changes in site space based on the modules' installation are also depicted.
- Crane mobilization events: In order to preform the modules lift (installation), the crane has to be mobilized from its current location to the new location to start the module lifting process. Once the crane mobilization event is fired by the construction site DES component, the pathfinding mechanism will calculate the safe, shortest path for the crane to mobilize to its target location. The pathfinding mechanism then sends the expected travel time and distance for this route back to the Site Construction Simulation Federate.
- Modeling changes in project's site space based on these simulated operations: The BGE based visualization screen depicts the changes in the site space (site layout) that occur because of adding modules or other permanent structures. Also it depicts changes in site space that occur because of changes in temporary facilities representations geometry. The decision maker controls temporary

facilities geometry on the visualization (BGE) screen through an interface as explained earlier in Chapter 3. Once a module or other permanent structure like a vessel or footing is added to its location onsite, the mesh generation mechanism changes the site space which was free and is now occupied by the module(s) or other permanent site structures from a Resources Admissible Site Space (RASS) to a Resources Forbidden Site Space (RFSS) as it was explained earlier in Chapter 3. The way the mesh generation mechanism does this in this case study will be more detailed later in this chapter.

It is worth mentioning that the changes in the state of the resources involved in the above mentioned operations can be represented in tabular data forms during the simulation run, however this does not provide an intuitive view of the simulation scenario and the logical sequence of the construction process. Tabular data forms representation is also limited in terms of representing the interaction and dynamic relationship between modules, space, and cranes as resources on the construction site.

As mentioned earlier, one of the challenges of construction simulation models is representing site space limitations (spatial data). A typical solution is by representing space as resource, yet this can be counter intuitive and is not easy to visualize by project team. For example, the simulation model, in this case, would only depict a crane as a resource with certain quantity without showing the crane maneuverability space or the crane routes. Also, the simulation of such a project might have flaws and errors especially when it comes to site installation sequence and the spatial conflicts that may arise from that. A 3D visualization component would be able to depict spatial conflicts better than textual output.

A flexible visualization component was sought after as the solution to detect spatial conflicts in the module assembly and site installation sequences of the modules. The HSV framework would be able to depict the spaces on the construction site as an initial step towards solving this case study’s contractor cranes’ tempo-spatial planning problem. Along the way, the framework will also show the module yard's occupied and free bays during the modules assembly operations as the project time (simulation time) advances in a more intuitive way. The pathfinding mechanism implemented in the visualization component of the framework will build on the framework’s ability to depict these changes in site space (model dynamic changes in site layout), and then find the cranes shortest paths along with the expected cranes mobilization activities durations on these paths.

In this implementation of the HSV framework, the game engine for Blender (an open source digital content creation environment) was utilized. The Blender Game Engine (BGE) provides a graphical interface that enables the development of interactive behaviors for 3D objects without the need for programming knowledge.

Two Visualization Federates are developed in this case, one for representing the utilization of space by different modules during their building processes on the module assembly yard, and the other for showing the on-site construction sequence of the modules and modeling site space changes on site, depicting also the dynamic changes in site layout. The PME (pathfinding mechanism) is implemented in the site visualization BGE. Both federates run simultaneously with other Simulation Federates and concurrently map the behaviors generated from the other federates to 3D worlds that represent the yard and the site. The architecture of these federates in relation to the whole federation is represented in Figure 4-3 and explained in the following sections.

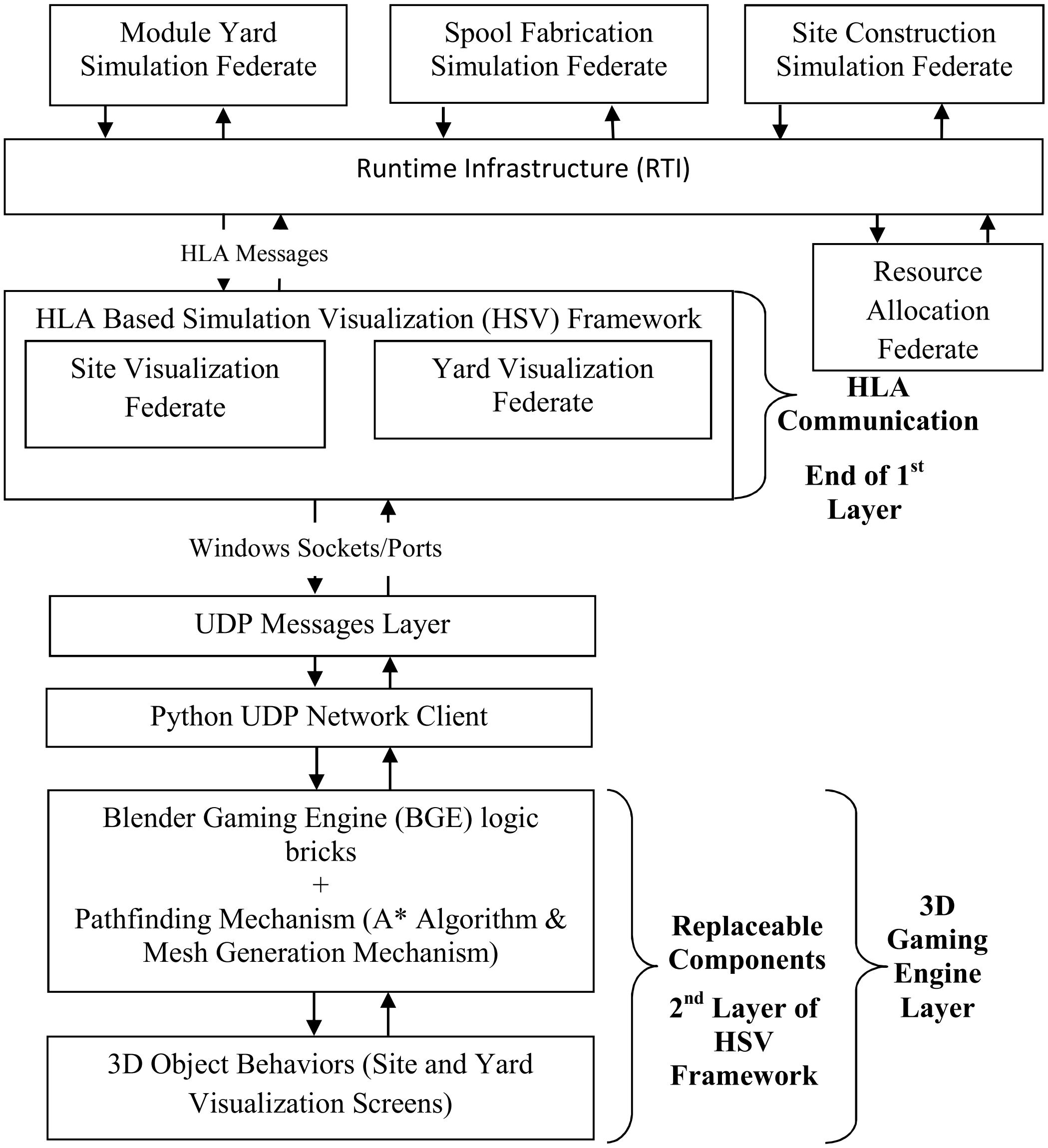

Module Yard Simulation Federate
Spool Fabrication Simulation Federate
Site Construction Simulation Federate
Runtime Infrastructure (RTI)
HLA Messages
Resource Allocation Federate
HLA Based Simulation Visualization (HSV) Framework
Site Visualization Federate
Yard Visualization Federate
HLA Communication
End of 1st Layer
Windows Sockets/Ports
UDP Messages Layer
Python UDP Network Client
Blender Gaming Engine (BGE) logic bricks
+
Pathfinding Mechanism (A* Algorithm & Mesh Generation Mechanism)
3D Object Behaviors (Site and Yard Visualization Screens)
Replaceable Components
2nd Layer of HSV Framework
3D Gaming Engine Layer

Figure 4-3 Framework used in simulation and visualization of pipe modules' assembly and their site installation

4.2.1 Main Elements of the Industrial Construction Federation

Communication between the BGE and the federation is enabled through a number of layers as shown in Figure 4-3. The first layer consists of the following:

- Simulation Federates: Where all the simulation of the various spool fabrications and spool assembly takes place. These federates produce the statistics needed by the decision-maker. Each federate represents not only a phase of the construction process but also work done in separate areas; namely the PCL Fabrication Shop (Spool Fabrication Federate), the PCL

Module Yard (Module Yard Simulation Federate) and the Construction Site (Site Construction Simulation Federate).

- Other Federates: The Resource Allocation Federate is responsible for allocating the various cranes to lift/move certain modules to their final location on site. Cranes are assigned to modules based on the cranes' availability and their fitness to handle certain module sizes and weights.
- Visualization Federates: Federates that read (subscribe to) the objects' (cranes and Modules) attributes and communicate these attribute values with a secondary layer of the framework that contains the visualization engines.

The second layer consists of the visualization engines, which are based on the BGE. Two visualization BGE screens are provided for the user; one showing installation of modules at their final location on site and the other showing module assembly progress with simulation time in module yard before transporting it to the site. The Visualization Federates provides the following to the decision-maker:

- The logical sequence of the modules' installation on site during the modules site installation operations. For example, the top module cannot be installed before the bottom one is installed, and a middle module cannot be installed after the left and right modules are installed. This provides a check for the logic of the construction schedule.
- Depiction of the dynamic changes in site layout and site spaces changes that result from the adding modules and other permanent structures to the site areas throughout the project's simulation run. This will be the basis of the framework's utilization in solving heavy lift cranes pathfinding problem in a dynamically changing site layout, as it will be explained later in this chapter.
- The progress of a module's assembly together with the bay it is assigned to, so the user can see bay occupancy during the course of the assembly process. This gives the user a depiction of the module yard bays' utilization and the distribution of the modules' assembly work load. The user can also pause the simulation and/or visualization to visually check the utilization of the bays at a given period of the project. Also, the user can make use of this visualization to test their construction schedule logic and make sure that there is no module installed before its predecessors are set in their final locations.

The developments of the visualization federates on the first layer of the framework in addition to the complete second layer of the framework together with the connections between the first and second layers were done as part of this research. Other developments (namely Site Construction Simulation and Yard Simulation federates) were developed by other members of the research team. The way that each of the two BGE visualizations is connected to its respective viewer federate will be explained in the following section.

4.2.2 Connection between BGE and Simulation

The two Visualization Federates subscribe to different attributes published by Simulation Federates during simulation.

The Site Viewer Federate subscribes to the following attributes:

- Module Name: this attribute helps identify the prototype 3D model that represents the module from a hidden layer inside the Site Visualization BGE.
- Module's Field Location: this attribute describes the state of the module on site and can take one of three values: "Delivered At Site", indicating that the module has arrived to the site; "In Lift", indicating that the module is currently being placed by a crane; and "At Set Point", indicating that the module is installed at its final location on site. These enumerations are predefined inside the federation's FOM and are produced during simulation by the site Simulation Federate to mark the location of each module instance at any point in of simulation time.
- Crane Availability State: As the site simulation run changes the state of each crane availability during the simulation run, the Site Visualization Federate reflects the changes to these attribute values in order to depict them to the decision-maker.
- Crane ID: Each of the 4 heavy-lift mobile cranes has a unique ID. Each crane is responsible for lifting certain modules based on its availability, capacity, proximity, and a module’s lifting specifications. A different 3D model represents each of the cranes. The visualization needs to call the right 3D model to represent a particular crane whenever the simulation triggers a scheduled event that utilizes this crane. Each of the 4 cranes might be at different locations of the site at any one moment, and each of these cranes’ movements needs to be planned simultaneously by the pathfinding mechanism built inside the site visualization BGE.

- Crane Location ID: Each of the predefined modules lifting locations has a unique ID. These locations are well known before the construction commences and marked on the site plans.

The Yard Viewer Federate subscribes to the following attributes:

- Module Name: this attribute identifies the prototype of the module from a hidden layer inside the Yard Visualization BGE.
- Module-Bay: This is published by the Module Yard federate and defines the bay allocated to each module inside the module yard at any point in time where the module assembly will take place.
- Module Progress: A percentage showing the completion of the module assembly.

Once the attributes are published, the viewer federates read them and sends a message containing the module unique identifier (module ID) and the module assembly's percent complete or its on-site state to the BGE. Same goes for the cranes, where each message sent from the Site Viewer Federate to the BGE has a crane unique identifier (crane ID). These messages contain the changes in the aforementioned attribute values of the modules and cranes as the site simulation proceeds. The execution of the BGE takes place as a completely independent execution thread and therefore, the communication between the viewer federates and the BGE is achieved through a network User Datagram Protocol (UDP), which provides language-independent communications between the .NET based HLA federation and the Python based BGE as explained earlier in Chapter 3. The Site and Yard Viewer Federates on the second layer of the HSV framework architecture shown, in Figure 4-3, act as both clients/servers to send/receive messages respectively to/from their respective BGE based visualizers. The client and server part on the BGE' side is based on a Python API, which sends and receives messages respectively from site or Yard Viewer Federates. The received messages are converted to BGE internal messages. The internal BGE messages are delivered to the corresponding 3D objects using BGEs through BGE graphical logic bricks as it will be explained in the following sections. This connection is a two-way communication (as explained earlier in Chapter 3) where each of the Viewer Federates and the BGE act as both servers and clients.

4.2.3 Mapping Simulation Product Classes Attributes to Visual Behaviors

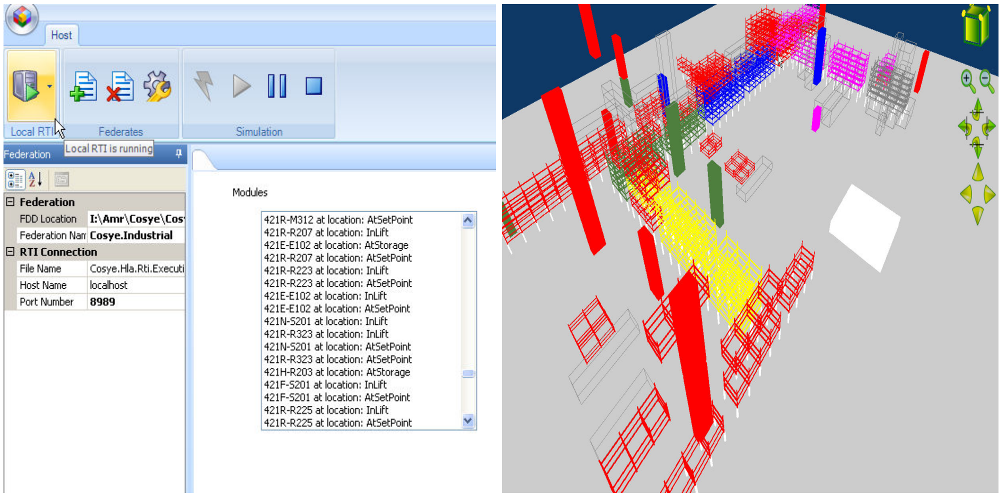
Product class inside the Site Simulation's FOM includes the final construction product models such as modules and vessels. The visual behaviors of the 3D objects are customized differently inside the Yard and Site Visualization BGEs. As the simulation runs the Site Viewer Federate reads the changes in the Module Field Location value for each module and sends messages through the UDP connection to the site visualization BGE. The BGE recalls the prototype of the module from a hidden layer and places it at its final location if the module's field location is set to be "At Set Point". This indicates that the module is lifted and installed in its final location based on the planned construction logic. On the assembly yard side, once a module is assigned to a bay through the Module-Bay attribute, this is reflected to the Yard Viewer Federate through the RTI and consequently, messages are sent from the Yard Viewer Federate to the yard visualization BGE stating the module's bay assignment. In addition, subsequent messages that update a module's assembly progress are also sent.

Once the yard visualization BGE receives these messages, it performs the following visual behaviors as the simulation time advances:

- Assigns a module to its bay in the yard 3D model once it is allocated one by the simulation
- Shows the change in modules' progress inside its respective bay as the simulation runs. This behavior is depicted by building the module prototype inside its bay till reaching the typical complete prototype of the module once the assembly progress is set to 100% by the Yard Simulation Federate.

The following Figures show screen shots of the viewer federates of the site and yard and their corresponding BGE visualizations. Figure 4-4(left) shows the messages in the list box which are sent to the BGE. These messages simply tell the BGE to recall a certain module using its module unique identifier, followed by its location on site. Once a module is "At Set Point" then it appears in its predefined location based on the design drawings. Figure 4-4(right) shows the BGE screen displaying modules at their predefined set points based on the messages received from the Site Viewer Federate.

The changes in site spaces geometry and site layout are automatically modeled on the BGE based site visualization screen based on these modules additions. Once a module (permanent structure) is added to an originally empty site space, this space classification changes using the mesh generation mechanism as it will be explained later in this chapter. These changes from RASS to RFSS will determine if the A* pathfinding algorithm can take this space in consideration when finding the shortest crane path once the site's DES triggers a resource mobilization event.



Federation

FDD Location	I:\Amr\Cosye\Cos
Federation Nan	Cosye.Industrial

RTI Connection

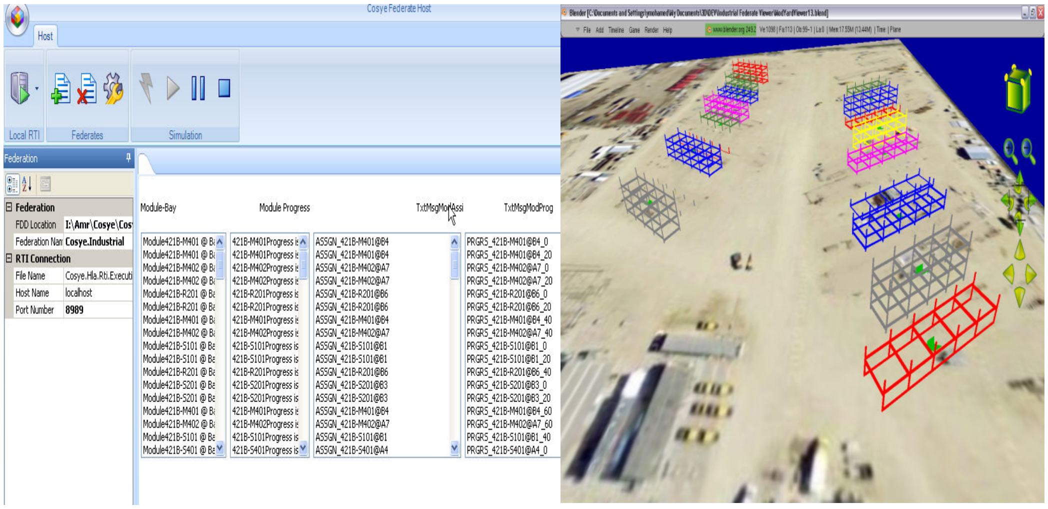
File Name	Cosye.Hla.Rti.Executi
Host Name	localhost
Port Number	8989

Modules

```
421R-M312 at location: AtSetPoint
421R-R207 at location: InLift
421E-E102 at location: AtStorage
421R-R207 at location: AtSetPoint
421R-R223 at location: InLift
421R-R223 at location: AtSetPoint
421E-E102 at location: InLift
421E-E102 at location: AtSetPoint
421N-S201 at location: InLift
421R-R323 at location: InLift
421N-S201 at location: AtSetPoint
421R-R323 at location: AtSetPoint
421H-R203 at location: AtStorage
421F-S201 at location: InLift
421F-S201 at location: AtSetPoint
421R-R225 at location: InLift
421R-R225 at location: AtSetPoint
```

Figure 4-4 Screens from Site Viewer Federate showing messages and Site Visualization using Blender Gaming Engine screen during simulation run

Figure 4-5(left) shows messages sent to the visualization BGE. One type of message in the list box called "Module-Bay" tells the visualization engine which module goes to which bay based on the simulation. The other message type contains the changes in percent completion of a module during the simulation run. The other message type is called "Module-progress" as explained earlier. Figure 4-5(right) shows the BGE yard screen assigning some modules to their respective bays using a real photo of the module yard and its bays as the texture for the site layout. One can observe that some modules are 100% complete while others are still building, indicating different progress percentage values received from the Yard Simulation Federate inside the "Module-Progress" message for each module.



Federation	
FDD Location	I:\Amr\Cosye\Cos
Federation Nam	Cosye.Industrial
RTI Connection	
File Name	Cosye.Hla.Rti.Executi
Host Name	localhost
Port Number	8989

Module-Bay	Module Progress	TxtMsgModAssi	TxtMsgModProg
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_0
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_20
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_0
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_20
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_0
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_20
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_40
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_40
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_0
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_20
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_40
Module421B-S201 @ Ba	421B-S201Progress is	ASSGN_421B-S201@B3	PRGRS_421B-S201@B3_0
Module421B-S201 @ Ba	421B-S201Progress is	ASSGN_421B-S201@B3	PRGRS_421B-S201@B3_20
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_60
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_60
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_40
Module421B-S401 @ Ba	421B-S401Progress is	ASSGN_421B-S401@A4	PRGRS_421B-S401@A4_0

Figure 4-5 Screens from Yard Viewer Federate and Yard Visualization BGE during simulation run

The way BGE receives these reflections and performs this animation through the simulation run will be explained in the following section.

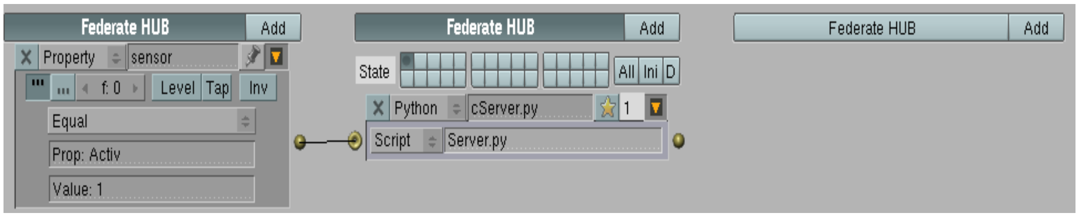
4.2.4 Generating Visual Behaviors in BGE and Modeling Site Spaces Based on Permanent Structures

Generation and customization of the behaviors of 3D objects inside BGE is done using graphical logic bricks. These logic bricks are built-in constructs in the Blender environment that allow developers create interactive game environments. Any object in the Blender environment can be equipped with these bricks, which allow it to respond to different events happening in the environment or generated by user activities. The logic bricks are broken down into three main types: “sensors”, “controllers”, and “actuators”. Sensors have different types and their main function is to sense a certain action or event that happens in the environment. Once a sensor is triggered, it sends a signal its controller. Controllers can be connected to different sensors and act as logical gates (e.g. “AND”, “OR”) to actuators. When all conditions of a controller are met, it fires one or more actuators. Actuators are the producers of visual behaviors in the game environment. A set of actuators owned by a certain 3D object will trigger it to produce different transformations according to the type of these actuators and the controller they are connected to. The following sections describe some of the logic bricks used to produce the visual behaviors of the site and yard visualization BGEs.

As an entry point to the game environment in Blender, a hidden object inside each of the BGEs acts as a server that receives messages during the simulation run and re-route them to trigger different logic bricks to produce the behaviors of 3D objects. A Python script on each of the two server objects does the following:

- Receives the messages from the respective viewer federate through a predefined port.
- Parses messages to pieces and sends them to the BGEs' logic bricks.

Figure 4-6 shows the logic bricks of the server object. In this case the server object is forced to signal a Python controller (i.e. execute the script in the controller) every certain number of animation frames. The script listens to the port, and routes the messages as described above.



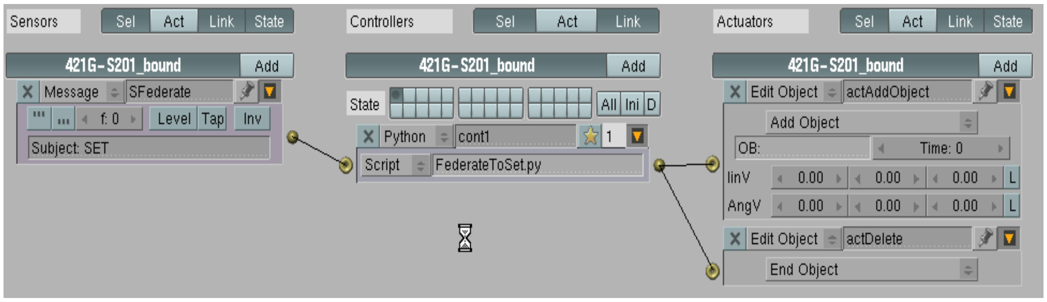
Property sensor
f: 0
Equal
Prop: Activ
Value: 1
Python cServer.py 1
Script Server.py

Figure 4-6 Logic bricks for the server object

A 3D game environment is created for each visualizer (i.e. site and yard visualizers). These environments contain different 3D objects. Some objects are static and they are mainly to enhance realism of the environment. Dynamic objects are equipped with logic bricks to react to simulation messages. The dynamic objects inside the site visualization BGE are:

- Module prototype models: these represent 3D models of the actual modules and were imported from CAD files of the project.
- Bound objects: these represent markers of the final set-points of the different modules and help manage the placement of module prototypes at their final locations on site.

The logic bricks of the bound objects are shown in Figure 4-7. A sensor is created to listen to messages with a subject equal to the string "SET". When such a message is sent to the object, it triggers two actuators. The first actuator adds the prototype model of the module to the scene at its final location. The second one deletes the bound object itself from the scene as it is no longer required.



Sensors: 421G-S201_bound
Message: SFederate
f: 0
Subject: SET

Controllers: 421G-S201_bound
State
Python: cont1, 1
Script: FederateToSet.py

Actuators: 421G-S201_bound
Edit Object: actAddObject
Add Object
OB:
Time: 0
linV 0.00 0.00 0.00 L
AngV 0.00 0.00 0.00 L
Edit Object: actDelete
End Object

Figure 4-7 Logic bricks of bound objects in the site visualizer

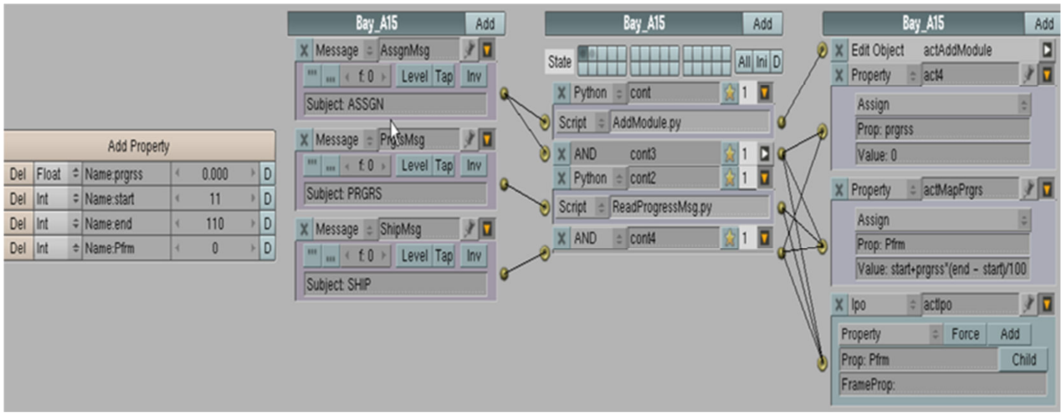
The dynamic objects inside the yard visualization BGE are:

- Module prototypes
- Bay objects

Figure 4-8 shows the logic bricks for a bay object. The object is equipped with three sensors for messages with subjects “ASSGN”, “PRGRS”, and “SHP”. These messages instruct the object to produce three different visual behaviors that represent 1) assignment of a bay to a module, 2) a certain progress of the module, or 3) shipping of a module out of the yard. An “ASSGN” message will trigger four actuators: the first one adds the module to the bay that is receiving this message, the second and third resets progress properties to 0, and the fourth resets an interpolated animation curve (IPO) to its original state, which represents 0 progress of the module.

A “PRGRS” message will first signal its controller to read the percentage of progress sent from the federate. Then, the message will trigger two actuators. The first actuator will calculate an animation frame to represent the percentage of progress sent. The second actuator uses the calculated frame number to change the look of the 3D object that represent the module. The change is based on a predefined IPO that represent different states of the module from 0% complete to 100% complete.

The last message (“SHIP”) triggers three actuators that reset progress properties and remove the 3D module prototype from the scene.



Add Property

Del	Float	Name:prgrss	0.000	D
Del	Int	Name:start	11	D
Del	Int	Name:end	110	D
Del	Int	Name:Pfrm	0	D

Message AssgnMsg f: 0 Subject: ASSGN

Message PrgrsMsg f: 0 Subject: PRGRS

Message ShipMsg f: 0 Subject: SHIP

Python cont 1 Script AddModule.py

AND cont3 1

Python cont2 1 Script ReadProgressMsg.py

AND cont4 1

Edit Object actAddModule

Property act4 Assign Prop: prgrss Value: 0

Property actMapPrgrs Assign Prop: Pfrm Value: start+prgrss*(end – start)/100

Ipo actIpo Property Prop: Pfrm FrameProp:

Figure 4-8 Bay Object logic bricks in the Yard BGE

4.2.5 Classifying Site Space based on Spaces' Occupation using Mesh Generation Mechanism

The mesh generation mechanism which was generically explained earlier in Chapter 3 is used here to classify the modeled site spaces into RASS and RFSS. Now that the way the site BGE visualization component has modeled the changes in site space that result from the simulation adding permanent structures to the site, these spaces has to be classified into RASS or RFSS as explained earlier. The way the mesh generation mechanism is also used to classify modeled site spaces based on changes in temporary facilities geometry changes is also explained. This classification is essential so the A* pathfinding algorithm operation, since the algorithm needs to identify which spaces to consider in its process of finding the shortest heavy lift cranes path, once a lift event is triggered by the site simulation component during the simulation run.

The initial mesh generation takes place at simulation time zero based on the initial site layout and is triggered once the visualization starts. The code is fired inside the BGE by connecting it to an Always sensor, meaning that the code is always fired as long as the simulation component is running to accommodate any changes done in the site layout by the simulation run.

The mesh generation code checks if there are any objects representing permanent structures or temporary facilities on this node itself or in the distance between this node and its neighbor nodes as explained earlier in Chapter 3. The code does this specifically in this case study by using a ray cast (Raycast) function, which is built into the BGE, meaning that an imaginary ray is cast from each node to its neighboring nodes. If the ray hits an object on the node itself or before it reaches the neighboring nodes, it means a structure, either permanent or temporary, is occupying this node. The way this is coded inside the BGE is as follows:

```
hit,p,n=ob.rayCast(a,node,0,"collision",0,1)
```

Where, hit is any structure occupying the neighboring node site space

p and n are the points from where the imaginary ray is cast and the imaginary ray normal direction

a is the neighboring node around the current node called “node”

“Collision” is a given property to any structure that occupies a site space and will change this site space from an RASS to an RFSS, making this site space un-processable by the A* algorithm. For example, every object in the site visualization screen that represents pipe modules carry a property called “Collision”, so the mesh generation mechanism can lable the space it occupies RFSS.

Now, as the simulation runs, the node mesh changes to accommodate for dynamic changes in site space and layout (permanent structures added to site spaces as explained earlier). This means that the node mesh update code discussed earlier in Chapter 3 need to be triggered every time the changes in site space geometry occur due to addition of permanent structures by the simulation. The updating of the node mesh algorithm is triggered through an Update message that is sent to the algorithm inside the BGE. Once a permanent structure is installed on those spaces, a message actuator is triggered sending an Update message in order to let the HSV framework update its already created node mesh. Figure 4-9 shows the logic bricks inside the BGE that trigger the Update message. The autonomy in creating RASS is now achieved inside the BGE.

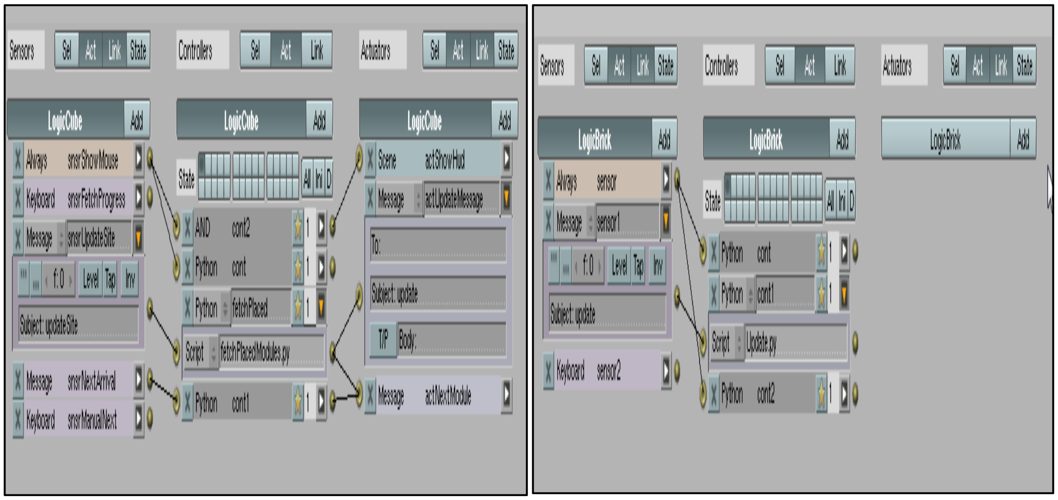


Figure 4-9 Logic bricks inside BGE site screen showing update message logical sequence

As shown in Figure 4-9 (left), the Update message is sent every time a structural element occupies a site space as the project's simulation time proceeds. Then, the Update message fires an Update.py module that calls the Update function inside the node mesh creation module and updates the RASS accordingly, as shown in Figure 4-9 (right).

On the other hand, temporary site facilities are not accounted for during the simulation. In other words, mobilizing or demobilizing a temporary facility is not an event that is triggered by the Site Construction Simulation Federate. Blocks—representing temporary construction facilities—can be controlled through keyboard interface by the decision-maker throughout the project's simulation lifecycle. The decision-maker can change these blocks' dimensions and their site locations on the site visualization screen during the simulation run (no need to interrupt the simulation). The user interacts with these blocks by simply moving them on the project plot with the arrow keys as the simulation is running. Every time the user changes the location of any of these temporary structures on the site plot displayed on the visualization screen, this automatically triggers an Update message similar to that produced when a permanent structural element is added to the site space by the simulation. This again triggers the node mesh update algorithm. Figure 4-10 shows how this is implemented in the BGE inside the visualization component of the HSV framework.

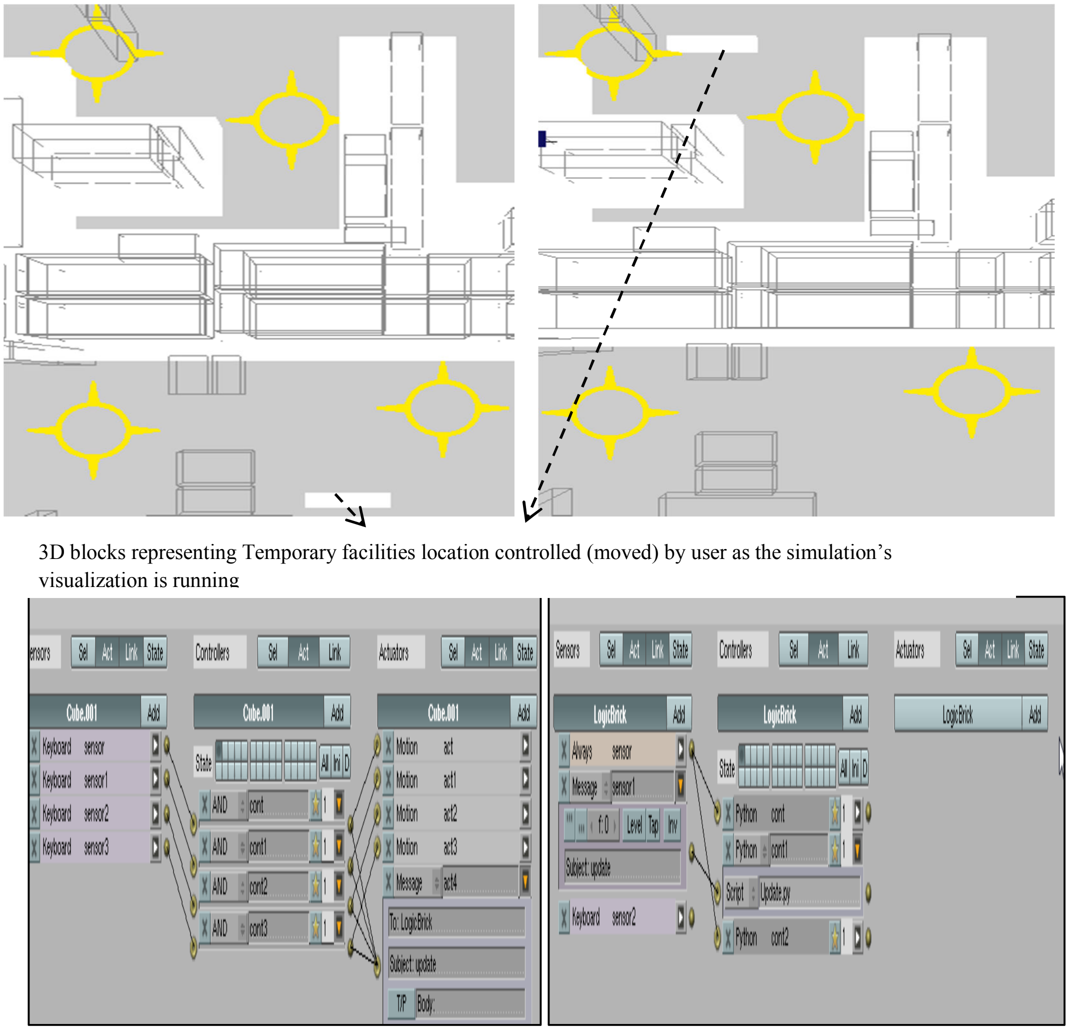

3D blocks representing Temporary facilities location controlled (moved) by user as the simulation's visualization is running

Figure 4-10 Logic bricks inside BGE site screen showing the update message triggering steps due to temporary structure block movement

As shown in Figure 4-10 (left), once the decision-maker moves the 3D blocks representing temporary site facilities by using the arrow keys, this triggers (with each move) an Update message. This message recalls the Update function that regnerates the node mesh to accommodate the different scenarios of spatial avaliability due to changes in temporary site facilities as shown in Figure 4-10 (right).

Section 4.2.7 will show screen shots and numerical examples from this case study that better explains the way these permanent and temporary facilities changes affect the site spaces and site layout, and the way the pathfinding mechanism accommodates these changes when calculating the cranes' shortest safe paths as the simulation is running.

4.2.6 Mapping Different Cranes' Simulation States into Visualization Behaviors and Application of Resources' Tempo-Spatial Planning Mechanisms

Now that the way that construction products simulation classes (modules) have their simulation states, that affect site's spatial data and layout, mapped to visual behaviors, it is time to do the same for simulation resources class (cranes) states. The Site Visualization Federates subscribe to the simulation's cranes object class attributes published by Simulation Federates during simulation. The Site Visualization Federate subscribes to the mobile crane object class attributes mentioned earlier in Section 4.2.2.

As the simulation runs, the Site Visualization Federate reads the changes in crane attribute (Crane ID and Location ID) values. It sends messages through the UDP connection to the site visualization BGE. The BGE recalls the 3D model that represents the crane from its current location onsite. Once the Location ID values for a certain heavy-lift crane change, the crane has to be mobilized from its current location to the new location to start the module lifting process. The pathfinding mechanism, which was explained in Chapter 3, starts to calculate the safe, shortest path for the crane to mobilize to its target location. The pathfinding mechanism then sends the expected travel time and distance for this route back to the Site Construction Simulation Federate.

This section will explain how the message is transferred inside the BGE once it is received from the simulation component, triggering the pathfinding algorithm. The way the BGE depicts the changes in the site space (site layout) by adding modules and other permanent structures were explained in earlier. Once a module is added to its location onsite, the site space which was free and is now occupied by the module(s) changes from a Resources Admissible Site Space (RASS) to a Resources Forbidden Site Space (RFSS), so that this space is not considered in mobilization planning. This change of the site's spatial state data is done through the site's mesh generation mechanism, which will be explained earlier in Section 4.2.5. Figure 4-11 shows the depiction inside the BGE of the modules (final construction elements) being installed at their final set points by the cranes.

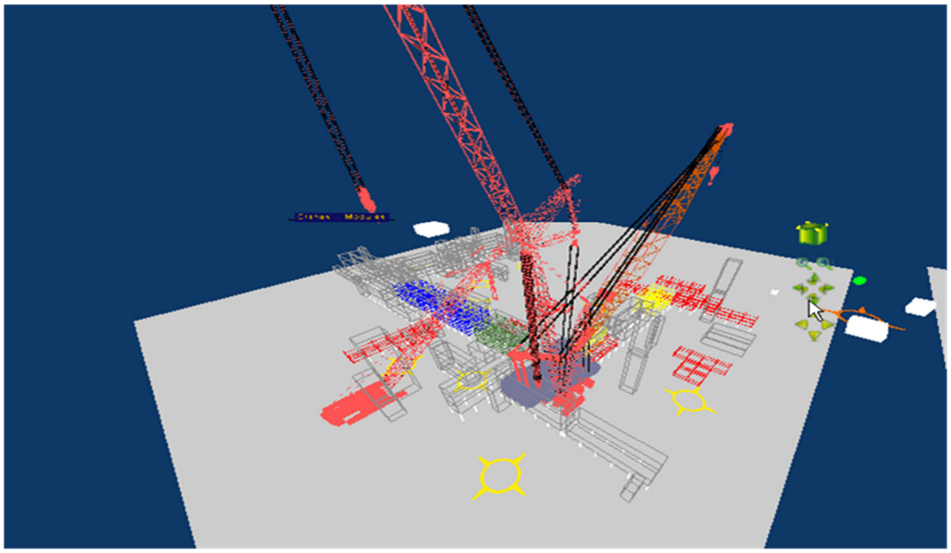
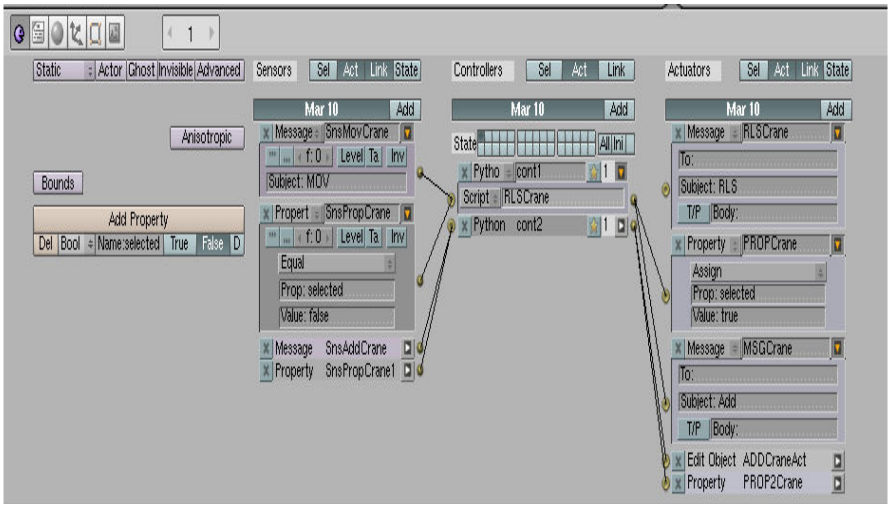


Figure 4-11 The BGE depicts the simulation of installing modules to their final set point using the heavy-lift cranes

The same logic brick mechanism that was explained in earlier is used here to create the visual behaviors for mobile cranes from their respective simulation states. These changes in crane simulation state also trigger the pathfinding algorithm for each crane mobilization. The server object first receives messages from the simulation. The message is time-stamped and fired when the project's simulation scenario schedules a lift activity involving a certain heavy-lift mobile crane. The parsed message then triggers the various crane visualization behaviors and the pathfinding mechanism.

The message carries the Crane ID and the Location ID. The server then extracts this parsed information and assigns it to different variables. Now the parsed Move Crane message goes through a series of built-in constructs (logic bricks similar to those used to create the visualization behavior of modules). This triggers various scripts including the A* algorithm to find the least-expensive, obstacle-free travel route from this crane's current Location ID coordinates to the target Location ID coordinates. Figure 4-12 shows the initial logic bricks that are put on each of the 10 marker 3D objects.



Bool Name:selected True False D

Message SnsMovCrane
f: 0
Subject: MOV
Propert SnsPropCrane
f: 0
Equal
Prop: selected
Value: false
Message SnsAddCrane
Property SnsPropCrane1

Pytho cont1 1
Script RLSCrane
Python cont2 1

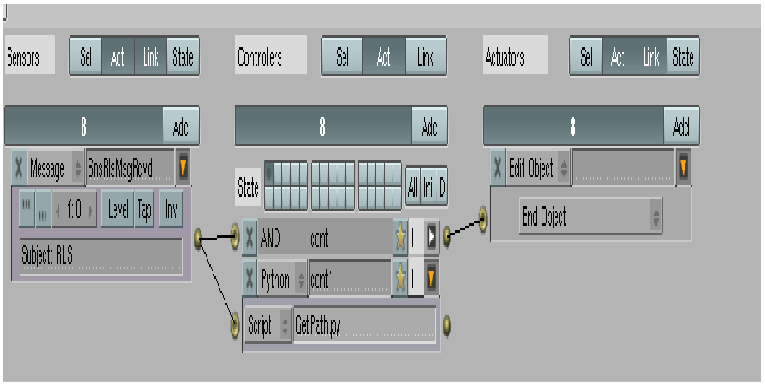
Message RLSCrane
To:
Subject: RLS
T/P Body:
Property PROPCrane
Assign
Prop: selected
Value: true
Message MSGCrane
To:
Subject: Add
T/P Body:
Edit Object ADDCraneAct
Property PROP2Crane

Figure 4-12 Screen from site visualization BGE showing the logic bricks that are put on object markers

The Move Crane message (MOV) that was sent by the Site Construction Simulation Federate is re-routed by the server to the object inside the BGE marking this Location ID. As shown in Figure 4-12, once the MOV message is received at the specified Location ID marker object, it starts a Python script (RLSCrane.py). This script does the following:

- It sends a Release Crane message (RLS), which is directed to the crane whose ID was mentioned in the MOV message. This message also contains the Location ID the crane should be heading to.
- It ends the crane's existence at its current location in order to move it to the new location using the route that will be decided by the A* algorithm.

Next the RLS message triggers the Python scripts containing the A* algorithm. This is done through a second layer of constructs (logic bricks) that are mounted on each crane 3D model. Figure 4-13 shows the second layer of the logic bricks implemented on each of the four cranes' 3D models inside the BGE.



Message SnsRlsMsgRcvd

Subject: RLS

AND cont

Python cont1

Script GetPath.py

Edit Object

End Object

Figure 4-13 Screen from BGE showing the second layer of the logic bricks implemented on each of the 4 cranes' 3D model

As shown in Figure 4-13, the RLS message activates the GetPath.py script that in turn calls the A* algorithm to execute (Figure 3-7 Grandchild Flowchart and Section 3.4). The script passes on the following parameters to the algorithm:

- Start: The current location of the crane model to which the RLS message was sent. The position is determined by the coordinates of the crane in the format (x,y,z) in the current site layout.
- Target: The Location ID the crane should be mobilized to in order to start the scheduled lift operation. The location is also given in the (x,y,z) coordinates format. The script returns the coordinates of the Location ID marker object that was part of the RLS message body.
- Nodemesh.nodes variable: The variable where all the information pertaining to the current node mesh covering the RASS is stored. The information is stored as sets of node coordinates that makeup the current node mesh, together with lists of coordinates for each node's neighboring nodes that form travelable edges. This variable decides on the RASS and constantly updates as the site layout changes.
- Own: The crane 3D model representing the current crane that is assigned for the scheduled lifting operation. Here the crane object itself is used as part of the built-in Blender ray-casting function. The pathfinding algorithm uses the crane object to cast imaginary rays before it moves from one node to a neighbouring one on the node mesh, in order to choose the obstacle-free path and make sure that there are no collisions with other models representing temporary facilities or permanent structures on the site layout. Together with the node mesh mechanism, this guarantees that the chosen shortest path is obstacle-free and ensures that the manoeuvrability distance of cranes is taken into consideration.

- Number of Steps before the Pathfinding Algorithm quits (variable): The maximum number of steps the A* algorithm will iterate through (move on nodes) before telling the decision-maker that there is no feasible route for the scheduled crane movement given the current site layout.

Once the algorithm is done running, it passes the following information back to the decision-maker as the simulation continues its run:

- Marks the shortest path of the given crane on the screen (Figure 4-14). The object representing the heavy-lift mobile crane with ID 2 moved from its start position to another using the green path which was generated by the A* algorithm to mark the resulting least-expensive, obstacle-free path.
- Sends the expected travel time (based on travel distance and average heavy-lift mobile crane speed) back to the simulation model.
- Confirms that the simulated crane movement took place using the least-expensive, obstacle-free route by messaging back the simulation.

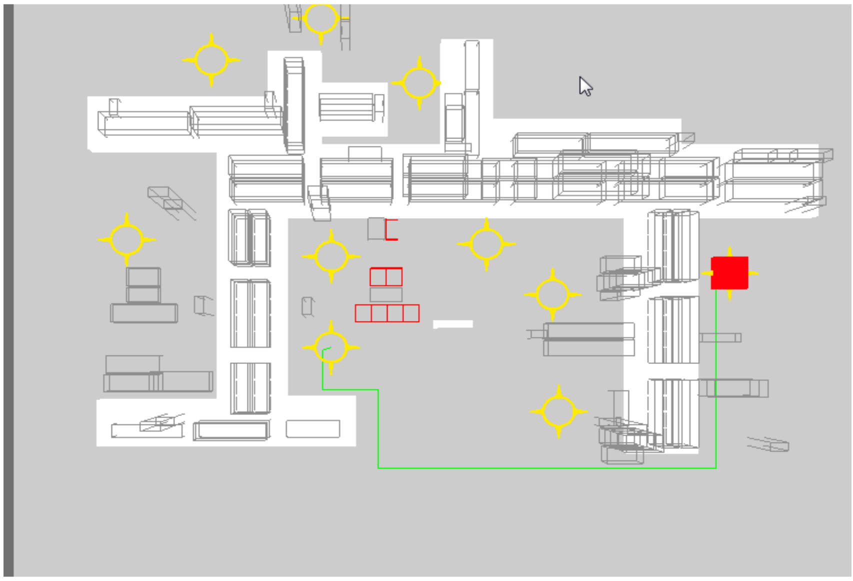


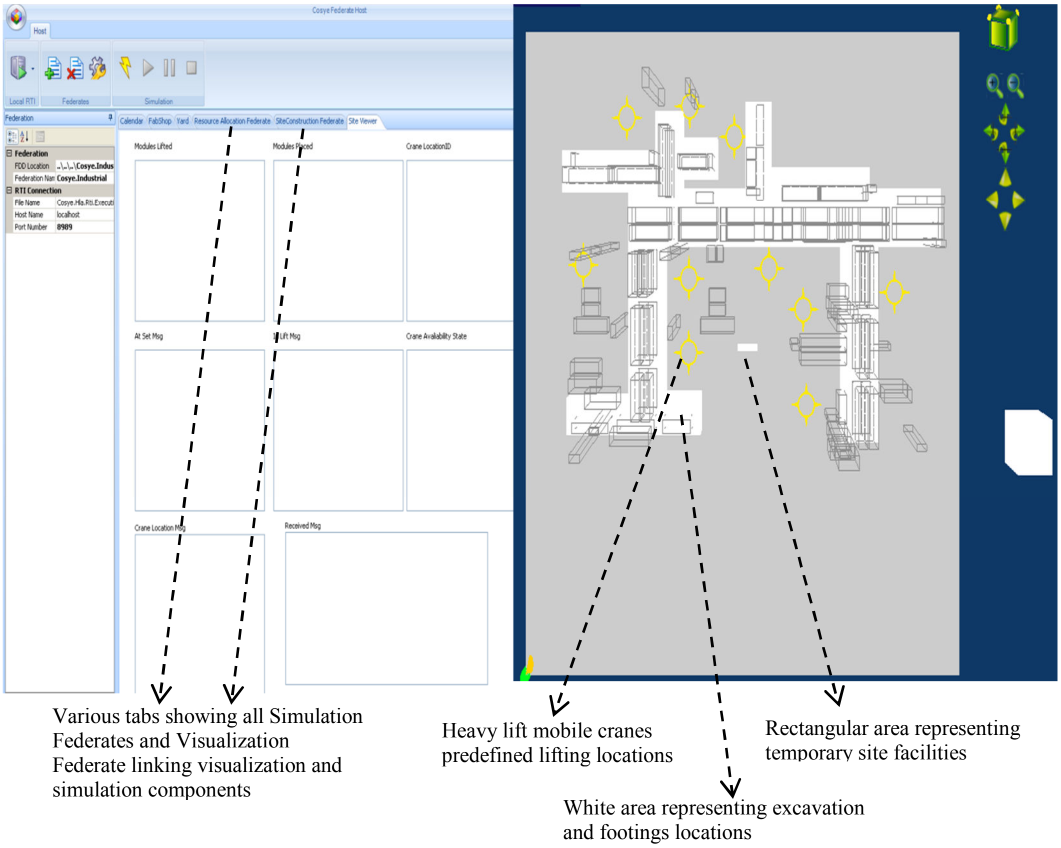
Figure 4-14 The path chosen by the A* algorithm marked on the site visualization user interface screen based on the current site layout

Next section will show screen shots and numerical examples from this case study that better explains the way the permanent and temporary facilities changes affect the site spaces and site layout, and the way the pathfinding mechanism accommodates these changes when calculating the cranes' shortest safe paths as the simulation is running.

4.2.7 Demonstrations and Screenshots

In this section, a comprehensive demonstration showing how the framework operates will be shown. This demonstration is a way of comprehending and summarizing all the information already explained throughout this case study chapter to show how the simulations, translations to visual behaviors, and the pathfinding mechanism interoperate to produce resource tempo-spatial management in this construction site. The example will be followed through a series of screen shots from the framework implementation in this case study with explanations.

Figure 4-15 shows the intial screens of the simulation and visualization before the start of the simulation run. One can notice the list of federates on the simulation user interface screen. The visualization screen also shows the initial site layout with the expected lift locations. Also, the area that will be excavated for footings is shown as being occupied by white blocks representing the foundation. This area constitutes a RFSS. A small rectangular representing temporary construction facilities is also shown. At simulation time zero the site is empty, i.e., there is no work progress.

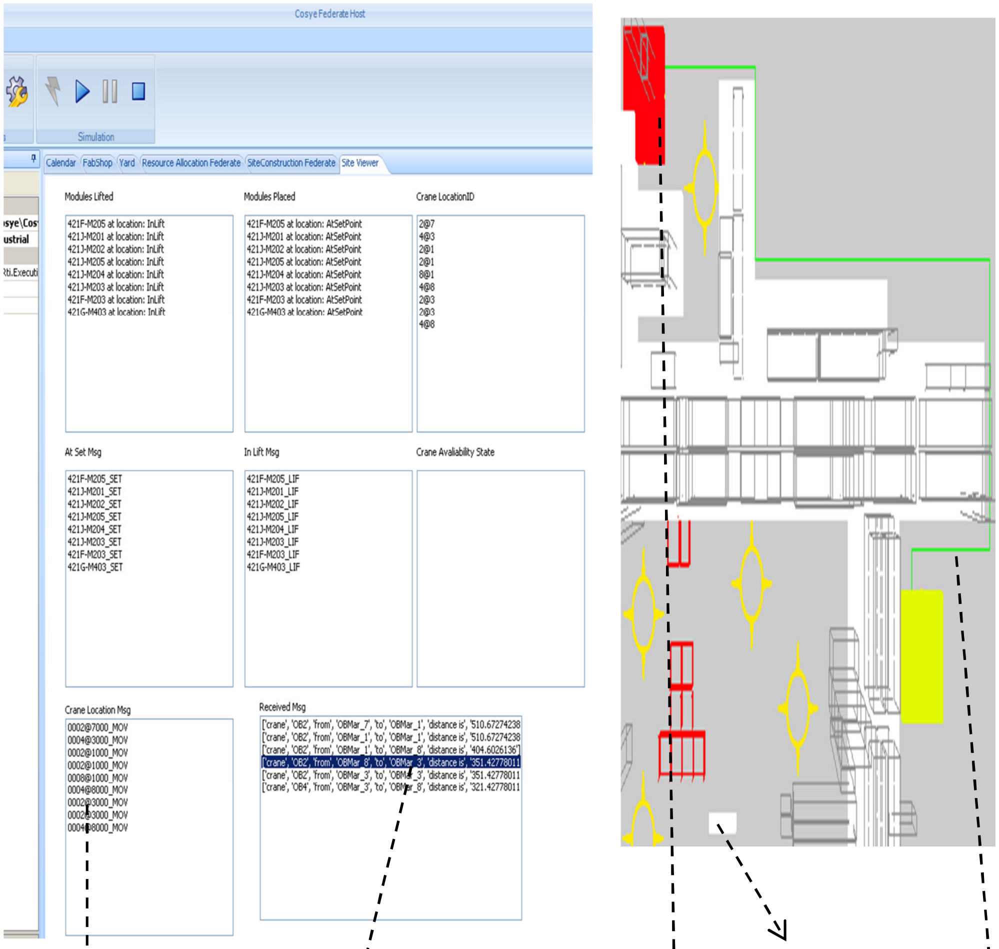


Federation
FDD Location ..\..\..\Cosye.Indus
Federation Nan Cosye.Industrial
RTI Connection
File Name Cosye.Hla.Rti.Executi
Host Name localhost
Port Number 8989

Cosye Federate Host
Host
Local RTI
Federates
Simulation
Federation
Calendar
FabShop
Yard
Resource Allocation Federate
SiteConstruction Federate
Site Viewer
Modules Lifted
Modules Placed
Crane LocationID
At Set Msg
Lift Msg
Crane Availability State
Crane Location Msg
Received Msg
Various tabs showing all Simulation Federates and Visualization Federate linking visualization and simulation components
Heavy lift mobile cranes predefined lifting locations
Rectangular area representing temporary site facilities
White area representing excavation and footings locations

Figure 4-15 Simulation Federates interface screen and BGE-based visualization screen at simulation time T=0

Once the simulation starts, the visualization component initializes concurrently generating the first mesh to cover the initial site layout. The initial site layout is divided into RASS or RFSS. Once the simulation starts, the Site Construction Simulation Federate starts to send messages to the visualization screen, which the visualization screen translates into visualization behavior and changes the site layout accordingly, updating the mesh covering the RASS. The A* algorithm built inside the visualization BGE analyzes each requested crane mobilization event and sends the simulation component a feedback containing its expected mobilization time and distance, ID of resource on the move, its original location, and final mobilization location; it also displays the route it will take on the visualization screen. All this information is calculated based on the layout of the site at the very moment the simulation component schedules the crane mobilization event. All this information is displayed to the decision-maker on the simulation user interface screen. The screenshots of the framework shown in Figure 4-16 illustrates this sequence of events.



Modules Lifted

```
421F-M205 at location: InLift
421J-M201 at location: InLift
421J-M202 at location: InLift
421J-M205 at location: InLift
421J-M204 at location: InLift
421J-M203 at location: InLift
421F-M203 at location: InLift
421G-M403 at location: InLift
```

Modules Placed

```
421F-M205 at location: AtSetPoint
421J-M201 at location: AtSetPoint
421J-M202 at location: AtSetPoint
421J-M205 at location: AtSetPoint
421J-M204 at location: AtSetPoint
421J-M203 at location: AtSetPoint
421F-M203 at location: AtSetPoint
421G-M403 at location: AtSetPoint
```

Crane LocationID

```
2@7
4@3
2@1
2@1
8@1
4@8
2@3
2@3
4@8
```

At Set Msg

```
421F-M205_SET
421J-M201_SET
421J-M202_SET
421J-M205_SET
421J-M204_SET
421J-M203_SET
421F-M203_SET
421G-M403_SET
```

In Lift Msg

```
421F-M205_LIF
421J-M201_LIF
421J-M202_LIF
421J-M205_LIF
421J-M204_LIF
421J-M203_LIF
421F-M203_LIF
421G-M403_LIF
```

Crane Avaliability State

Crane Location Msg

```
0002@7000_MOV
0004@3000_MOV
0002@1000_MOV
0002@1000_MOV
0008@1000_MOV
0004@8000_MOV
0002@3000_MOV
0002@3000_MOV
0004@8000_MOV
```

Received Msg

```
['crane', 'OB2', 'from', 'OBMar_7', 'to', 'OBMar_1', 'distance is', '510.67274238
['crane', 'OB2', 'from', 'OBMar_1', 'to', 'OBMar_1', 'distance is', '510.67274238
['crane', 'OB2', 'from', 'OBMar_1', 'to', 'OBMar_8', 'distance is', '404.6026136']
['crane', 'OB2', 'from', 'OBMar_8', 'to', 'OBMar_3', 'distance is', '351.42778011
['crane', 'OB2', 'from', 'OBMar_3', 'to', 'OBMar_3', 'distance is', '351.42778011
['crane', 'OB4', 'from', 'OBMar_3', 'to', 'OBMar_8', 'distance is', '321.42778011
```

Temporary site facility at this simulation time located here

The visualization component sends the simulation a feedback (during the simulation run) that a path was found and crane travel distance of **351.4 m** is expected

Green path marked by A* algorithm showing shortest obstacle free path to move crane 2 from location 8 to location 3

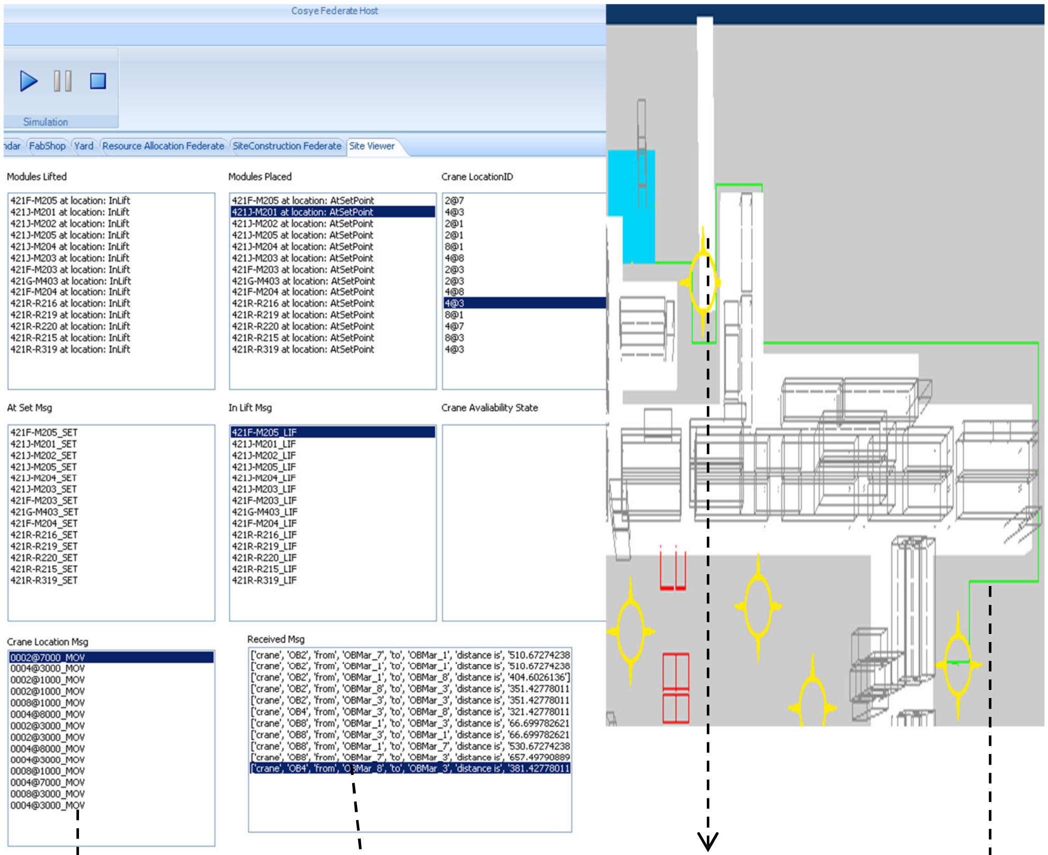
Simulation message to move Crane 2 to Location 3

Red object representing Crane 2 moving to Location 3

Figure 4-16 The A* algorithm decides on least-expensive, safe travel distance for Crane 2 avoiding temporary and permanent structures on the current site layout

As shown in Figure 4-16, a message was sent by the Site Construction Simulation Federate to move Crane 2 from Location 8 to Location 3 in order to start a lift operation. Crane 2, represented by the red cube on the Visualization Federate, was already at Location 8 at an earlier simulation time. Based on this layout, the A* algorithm shows the shortest path to move Crane 2 from Location 8 to Location 3 in green, and the visualization completes the movement operation and sends its feedback to the simulation confirming that Crane 2 can move from Location 8 to Location 3 and the least-expensive, obstacle-free route was found, its expected distance being 351.4 meters based on the current site layout.

What if the user changes the location and size of a temporary construction facility at any stage of the project? This can be simply tested by moving/resizing the white rectangular representation and moving it on the site layout (shown on Figure 4-16) as indicated earlier. To demonstrate, a temporary construction facility will block this chosen path between Locations 8 and 3 as shown in Figure 4-17 below. As the simulation runs, it schedules another crane (Crane 4) to move between Locations 8 and 3. Crane 4 is supposed to follow the same path that was followed earlier by Crane 2 to move between these two locations (path marked by green line in Figure 4-16); however, now there is a temporary structure blocking this path changing part of the site space from RASS to RFSS. Therefore, Crane 4 (whose dimensions are not that different from the dimensions of Crane 2) is given a different path by the A* algorithm to move between the same locations. This is demonstrated in Figure 4-17 below.



Modules Lifted

```
421F-M205 at location: InLift
421J-M201 at location: InLift
421J-M202 at location: InLift
421J-M205 at location: InLift
421J-M204 at location: InLift
421J-M203 at location: InLift
421F-M203 at location: InLift
421G-M403 at location: InLift
421F-M204 at location: InLift
421R-R216 at location: InLift
421R-R219 at location: InLift
421R-R220 at location: InLift
421R-R215 at location: InLift
421R-R319 at location: InLift
```

Modules Placed

```
421F-M205 at location: AtSetPoint
421J-M201 at location: AtSetPoint
421J-M202 at location: AtSetPoint
421J-M205 at location: AtSetPoint
421J-M204 at location: AtSetPoint
421J-M203 at location: AtSetPoint
421F-M203 at location: AtSetPoint
421G-M403 at location: AtSetPoint
421F-M204 at location: AtSetPoint
421R-R216 at location: AtSetPoint
421R-R219 at location: AtSetPoint
421R-R220 at location: AtSetPoint
421R-R215 at location: AtSetPoint
421R-R319 at location: AtSetPoint
```

Crane LocationID

```
2@7
4@3
2@1
2@1
8@1
4@8
2@3
2@3
4@8
4@3
8@1
4@7
8@3
4@3
```

At Set Msg

```
421F-M205_SET
421J-M201_SET
421J-M202_SET
421J-M205_SET
421J-M204_SET
421J-M203_SET
421F-M203_SET
421G-M403_SET
421F-M204_SET
421R-R216_SET
421R-R219_SET
421R-R220_SET
421R-R215_SET
421R-R319_SET
```

In Lift Msg

```
421F-M205_LIF
421J-M201_LIF
421J-M202_LIF
421J-M205_LIF
421J-M204_LIF
421J-M203_LIF
421F-M203_LIF
421G-M403_LIF
421F-M204_LIF
421R-R216_LIF
421R-R219_LIF
421R-R220_LIF
421R-R215_LIF
421R-R319_LIF
```

Crane Avaliability State

Crane Location Msg

```
0002@7000_MOV
0004@3000_MOV
0002@1000_MOV
0002@1000_MOV
0008@1000_MOV
0004@8000_MOV
0002@3000_MOV
0002@3000_MOV
0004@8000_MOV
0004@3000_MOV
0008@1000_MOV
0004@7000_MOV
0008@3000_MOV
0004@3000_MOV
```

Received Msg

```
['crane', 'OB2', 'from', 'OBMar_7', 'to', 'OBMar_1', 'distance is', '510.67274238
['crane', 'OB2', 'from', 'OBMar_1', 'to', 'OBMar_1', 'distance is', '510.67274238
['crane', 'OB2', 'from', 'OBMar_1', 'to', 'OBMar_8', 'distance is', '404.6026136']
['crane', 'OB2', 'from', 'OBMar_8', 'to', 'OBMar_3', 'distance is', '351.42778011
['crane', 'OB2', 'from', 'OBMar_3', 'to', 'OBMar_3', 'distance is', '351.42778011
['crane', 'OB4', 'from', 'OBMar_3', 'to', 'OBMar_8', 'distance is', '321.42778011
['crane', 'OB8', 'from', 'OBMar_1', 'to', 'OBMar_3', 'distance is', '66.699782621
['crane', 'OB8', 'from', 'OBMar_3', 'to', 'OBMar_1', 'distance is', '66.699782621
['crane', 'OB8', 'from', 'OBMar_1', 'to', 'OBMar_7', 'distance is', '530.67274238
['crane', 'OB8', 'from', 'OBMar_7', 'to', 'OBMar_3', 'distance is', '657.49790889
['crane', 'OB4', 'from', 'OBMar_8', 'to', 'OBMar_3', 'distance is', '381.42778011
```

Simulation now scheduling Crane 4 to Location 3

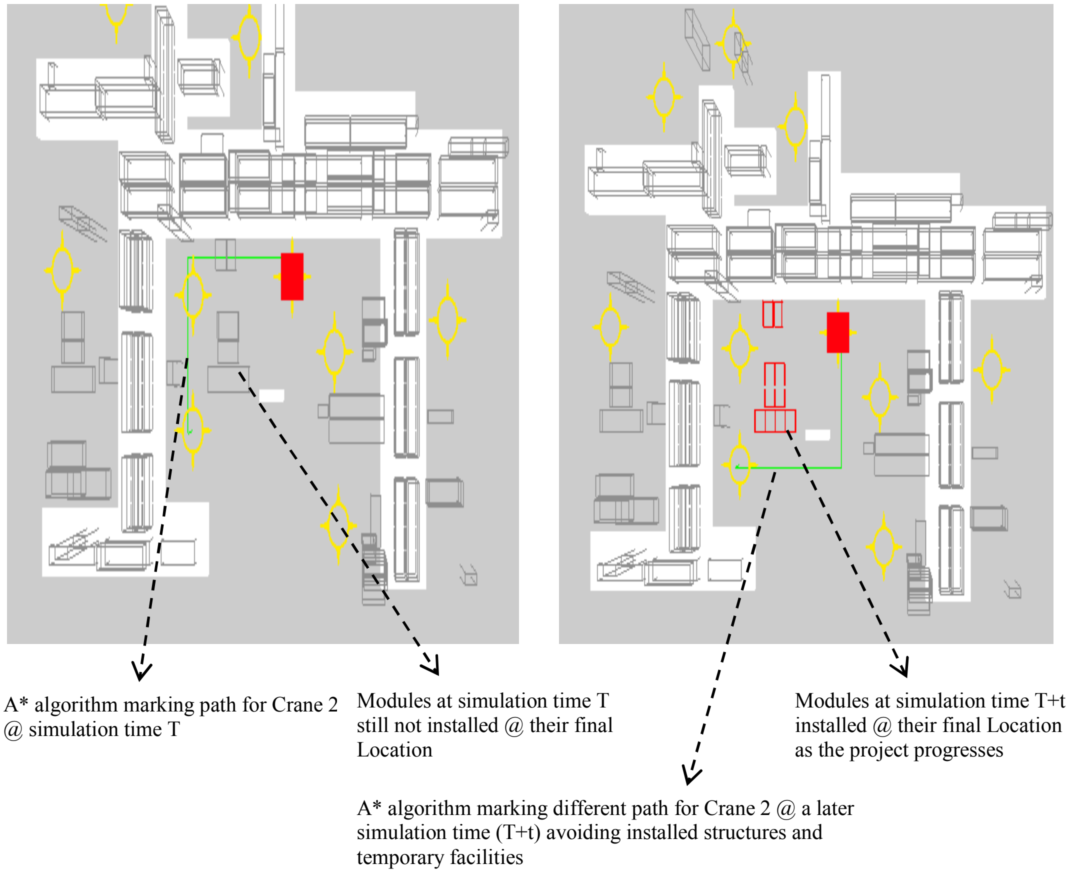
Temporary site facility at this simulation time moved here by the decision-maker

The Visualization Component sends the simulation a feedback (during the simulation run) that a path was found but now the travel distance is **381 m** instead of **351 m** due to the temporary facility that the decision-maker has added

New green path marked by A* pathfinding algorithm showing the new path between Locations 3 and 8

Figure 4-17 Screen shots showing the A* algorithm at a later simulation time deciding on shortest safe travel distance for Crane 4 after a change in site layout due to an addition of a temporary construction facility

Although Figure 4-17 shows only an extra 30 meters of travel onsite, which might not be a significant increase in the crane travelling distances, however this is only for demonstration of the resource pathfinding that can be done by this framework using its implemented pathfinding mechanism.



A* algorithm marking path for Crane 2 @ simulation time T

Modules at simulation time T still not installed @ their final Location

Modules at simulation time T+t installed @ their final Location as the project progresses

A* algorithm marking different path for Crane 2 @ a later simulation time (T+t) avoiding installed structures and temporary facilities

Figure 4-18 Crane 2 route (in green) moving between Locations 2 and 7 at simulation time T (left) and at simulation time T+t, where (T+t >T) (right)

Figure 4-18 demonstrates the way the A* algorithm accounts for changes in site layout due to installation of permanent structural elements to the site space by the simulation run. The green path the A* algorithm gave for Crane 2 movement between the same lift locations was different when the events were scheduled at simulation times T and T+t, respectively (where T+t>T). This is because as the simulation reached simulation time T+t and the project progressed, the simulation had already installed several modules (permanent structural elements) in place as shown on the site layout (Figure 4-18). All this is done conccurently as the simulation is running.

4.3 Framework's Visualization Screen Conceptual Design Based on Visual Analytics and Display Design Principals

Conceptual design of the framework's visualization screens used to solve the heavy lift cranes pathfinding problem in the Ft. Saskatchewan site had to take place before and during its technical development. In this section of the case study chapter, the guides used in developing the visualization component of the HSV framework implemented in this case study are discussed. These guides were discussed earlier in Chapter 2. It is also worth mentioning that this is the first time in construction research that visual analytics and display design guides are used in the conceptual design of a visualization based mechanism which is used in solving a construction problem. The guides that Russell et al., 2009 set for visual analytics design explained earlier in Chapter 2 were utilized in the conceptual design phases of the visualization component of the HSV framework implemented in this case study. Then, the 13 principles of human perception applied to display design (Chapter 2) were utilized in the translations of the chosen simulation's object classes attributes' states into corresponding visualization behaviors.

4.3.1 Site Visualization Screen: Conceptual Design In Light Of Visual Analytics Design Fundamentals

To account for heterogonous user audiences that will use this framework for heavy lift crane movement decision making in this case study, it was decided to design the visualization in a way to make it simple to comprehend by a heterogeneous audience of decision-makers using a common visual representation. This was achieved by showing the complete site layout as the terrain or global environment of the visualization. It is more of a map-like representation of the site layout, where the dynamic changes, which are triggered by the DES component of the framework, are mapped directly on this site layout map to reflect the changes in site space. Since humans read and comprehend with maps in their day to day activities, it was thought that the simplest way to depict changes in site layout would be through the method mentioned above. In addition, the least-expensive, safe cranes paths which are defined by the pathfinding mechanism implemented in the visualization component of this framework are marked on this dynamically changing site layout with eye-catching colour in a similar way that a GPS device marks a car route to its destination. For decision-makers who would rather see these shortest paths expressed quantitatively in the form of expected travel distances and time, the visualization's pathfinding mechanism send its output, which consists of analysis of the various cranes' paths, to the simulation component concurrently as it is running. This also enables the decision-maker to see the results in a tabular format not just visually. The visualization screen also has a visual data exploration feature, adding more simplicity to the users' interaction with the visualization screen. This simplicity can make it more comprehensive for a wider spectrum of heterogeneous decision-makers in the construction industry. It was also decided to choose 3D representation for the visualization space as it is most appealing and intuitive for decision-makers. 3D visualization can also be easily adjusted to 2D visualization.

4.3.1.1 Site Visualization Screen: Data Representation Based On Visual Analytics Purpose

As mentioned in Chapter 2, the first step to design the visualization screen based on visual analytics, is understanding the reasons behind the framework's visualization development. This was the key to choosing the published site simulation's resources and products object classes attributes' values that were translated to visual behaviors by the Site Visualization Federate.

Table 4-1 shows the reasons behind the development of the framework and the areas in which it is expected to be used as a visual analytics system. The second column of the table describes the simulation data (objects classes' attributes) that needs to be represented by the visualization to provide the analytical perspective to the decision-maker. These visual representations will aid the decision-maker in achieving the visual analytics purposes in the first column.

Visual Analytics Purpose	Generic Simulation Data To Be Represented By Visualization Component
Depicting construction logic and flaws in simulation process	Simulation's products object classes' attributes (e.g. Modules completion date, Modules installation date, Module lift state)
Depicting changes in site spaces throughout the project and dynamic site layout changes	Simulation's resources classes' location attributes and Product classes' attributes (e.g. Crane location, module set point location, Crane state attributes, Module field location, Module state)
Finding the shortest cranes paths together with expected cranes onsite mobilization times using the pathfinding mechanism implemented in HSV framework visualization component	All the simulation's object classes' attributes that makes the visualization depict (predict) the changes in site layout

Table 4-1 conceptually formulating the purpose of visual analytics in HSV framework and the simulation data to be visually represented to achieve it

4.3.1.2 Site Visualization Screen: Designing interactions and Visual Representations

Now that the data to be represented based on the visual analytics purpose has been chosen, it was time to decide on the way that this simulation object classes' attribute values will be transformed into visual behaviors. A dilemma for representing these qualitative attributes, like changes in site space for example, was the level of granularity or level of details it will be visualized at. Since the purposes (goals) of this visual analytics (shown in Table 4-1) are at the overall site level and non-detailed categorical construction processes level, (both at a lower level of details), it was decided to represent the overall site at a lower level of details. For example, show the overall changes in a site's spaces on a visual representation of the overall site layout.

For the remaining aspects of transforming the chosen represented simulation data to visualization behaviors, the 13 design principles of display design mentioned earlier will be utilized as guides for creating these visual behaviors. This will be explained in section 4.3.2 of this chapter.

Finally, other visual analytics guides that were used in the visualization component conceptual design are shown in Table 4-2.

Visual Analytics Guide	Application in HSV Visualization Display Design
Follow conventions and good practices of graphics data which have formed people basic graphics literacy over the years (Schmid,1978)	For example, the least-expensive, safe resource paths were represented only by a single simple straight line on the site layout visualization screen as they are more conventional for viewers For example, Resources to be represented on site by simple geometrical 3D shapes
Address the various visual analytics purposes on one screen rather than multiple screens; this reduces the time needed to analyze multidimensional data as it prolongs users' patience keeping decision-makers more attentive and engaged (Cawthon and Moere, 2007)	In the visualization component's design, the various visual analytics depictions that the visualization screen is trying to show (Table 4-1) can be achieved through a single construction site visualization screen rather than multiple screens. If construction projects are made up of unconnected locations, then each location's simulation visualization can be represented by its own screen as it will be shown in Chapter 3.
Use interaction features that allow users to interact with data's visualization (Russell et al.,2009)	Features were implemented in the visualization component to allow users to interact with the simulation visualization screens by changing /adding or removing temporary site facilities to site layout during the run of the simulation itself as it will be shown in Chapter 4.

Table 4-2 Visual analytics design guides and how they were applied in the conceptual development of visualization component's screens

4.3.2 Site Visualization Behaviors Choices In Light Of the 13 Display Design Principles

The 13 display design principles explained earlier in Chapter 2 were conceptually used as guides to map the various chosen site DES object classes' attributes states into corresponding visual behaviors. This section contains a briefing on the way these principles were utilized in the translation processes of the simulation's cranes and products' attributes values into visual behaviors.

The principles were broken down into 4 categories and explained from a generic perspective in Chapter 2. The way that they are applied to map the various chosen site DES object classes' attributes states into corresponding visual behaviors will be explained here following the same organization that was used to explain the principles earlier in Chapter 2.

A- Perceptual Principles:

1. The visualization legibility principle: Based on this principle, the following was taken into consideration when designing the HSV visualization component's display:
    - The display clearly mapped abstracts of the changes in represented site's construction DES object classes' attributes' values into comprehendible visual behaviors.
    - The decision-makers looking at the site visualization screens were able to comprehend the meanings of those visual behaviors on their own or with minimal explanation.
    - The users of the framework were able comprehend the reasons for developing this visual analytics HSV framework (solving the heavy cranes pathfinding problems and modeling changes in site's spatial data) on their own or with minimal explanation.

    As mentioned earlier, a resource's shortest distance is represented on the site visualization screen by a line similar to that which is drawn on a car GPS to show the route to its destination. Using common sense, a decision-maker will understand that the line shown on the current site layout display screen represents the requested resource's shortest travel distance. That is because it is well established on various pathfinding devices that routes are represented by coloured lines. This is an example of how the site screen visualization used here was legible.

2. The principle of avoiding absolute judgment: Limitation of judgment should be avoided when representing the simulation object classes' models inside the visualization. According to this principle, the 3D visualization models representing similar entities of the simulation's products or cranes object classes were not coloured using different degrees of the same colour or different colours which are indistinguishable. For example, the different 3D models which represent the different pipe modules and cranes were

represented by independent colours, not degrees of the same colour. For example, red, green and blue are independent colours.

Also, the changes in an object class during the simulation run was not represented through a variable that has unlimited levels. For example, the placement of modules at a set point by the simulation model is represented as disappearing and re-appearing of the modules in its place, and not by mere change of colour of this module. The same can be said for the depiction of a simulation's cranes mobilization event. The cranes' mobilizations from origin to destination are depicted by appearing at a destination site position and disappearing from its original site position. This is in harmony with this principle since the appearing –disappearing action has only 2 levels or states (appearing state and disappearing state) rather than representing the above mentioned simulation events with a variable like colour which has unlimited levels.

On the other hand, when the depiction of modules building simulation events takes place, lines are added to the 3D models representing these modules on the visualization screen, depicting the modules' development states. This depiction does not allow the user to judge exactly the development stage of the module and this is in violation of the principle of avoiding absolute judgment limits.

3. Top-down processing or the principle of immediate context of a display: It implies that all the similar simulation events involving various object classes were grouped and translated to similar visualization behaviors. For example, in the HSV framework's implementation in the industrial construction case study, the decision-maker will expect that all the simulation events of modules' installation to their respective set points, to be represented by the disappearing and re-appearing visualization behavior of the modules' 3D representations. Also, similar simulation events involving different cranes were mapped into similar visualization behaviors.

4. The principle of redundancy gain: In this case study; when the simulation triggers an event which sends a module to its set point, it is mapped into two simultaneous visualization behaviors on the visualization display; the appearing and disappearing of the module representations in the visualization, simultaneously with a crane's representation movement to the lifting location on the site layout. This means that two different visualization behaviors represent a single simulation event. This is not considered a repetition but it is an application of the redundancy gain principle to stress the simulation event translation inside the decision-maker's mind.

   Another example of the use of the redundancy gain in the above mentioned implementation is the visualization of the simulation states of modules being built in the module yard. To represent a module's building as being completed (progress=100%), two visual representations take place; first the lines are added to the module's 3D models to represent the 100% completion of the module's building, then the completed module disappears from the bay. This is another example of the application of the redundancy gain principle in translating a single simulation state to two visualization behaviors inside the HSV framework visualization design.

   It is worth mentioning that not all the simulation events translations into visualization behaviors followed the principle of redundancy gain. Some simulation events were translated to a single visualization behavior. These simulation events translations are not in accordance with the principle of redundancy gain.

5. Principle of discriminability: Although the different instances of 3D representations experience the same visualization behavior throughout the simulation run, their different colours make them distinctive. Although these instances are distinguishable through their different colors, yet that is not enough to satisfy this discriminability design principle. The design of the simulation visualization did not strictly follow this principle. Hence, the choice of object classes' 3D representations colours does not contradict

with the absolute judgment principle (principle Number 2) discussed earlier.

B- Mental Model Principles:

6. Principle of pictorial realism: For example, the simulation's crane object class visualization prototypes were built in accordance with the concept of pictorial realism rather than representing these cranes in an iconic way. This is one of the strengths of this HSV framework; it allows typical realistic representation of simulated object classes instead of the old iconic representation of simulated elements. However, it is fair to say that some represented object classes in the HSV framework were represented in an iconic way. The author thought that such higher level of details in the object classes' representations in the visualization was not significant to the decision-makers.

7. The principle of moving parts: When a certain object is transformed during a construction process, either this movement is a translation or rotation, the simulation of this process should have an object class that mimics how this object behaves in a real construction site. The visualization of this simulated process will in turn have a model that represents this object class, and the transformation(s) of this representation should correspond to its simulated movement. For instance, in the HSV framework, the simulation events of the cranes' mobilization from a lift location to the other on site constitute a form of object translation. On the site visualization screen, the shortest route that this crane model will follow (Chapter 4) is published through the pathfinding mechanism built inside the visualization component of the framework, and the crane model is translated to its destination lift location on this path.

C- Principles Based on Attention:

The HSV framework allows for the three types of attentions discussed in Chapter 2 to be present in the site simulation visualization concurrently. When it comes to Selective Attention, the decision-maker can use the site visualization screen to only concentrate on the depiction of a certain simulated construction operation. Focused Attention is also another point of strength of this framework, since the framework gives the decision-maker the ability to view the processes on single or multiple visualization screens, building on the focused attention of the decision-maker. For example in this case study, separate visualization screens were used to depict the simulated module yard processes and site construction processes. Also, in the HSV framework's implementation in this case study; although different viewers were used to depict various construction operations taking place at different geographical locations of the project, however processes taking place at the same location are depicted on the same viewer. This builds on the decision-makers' Divided Attention.

The following 4 attention based principles explained earlier in Chapter 2, were taken in consideration when designing the visualization screens of the HSV framework used in this case study:

8. Minimizing the information access cost principle: It was mentioned earlier that the multiple-visualization screens inside the HSV framework capitalize on both the focused attention and divided attention of the decision-maker. Yet, according to this principle it is always better for the visualization component design to keep the number of those screens as minimal as possible. Usually, 2 to 3 screens at most are used for each project's simulation visualization. This is done to minimize the cost to move the decision-maker's selective attention between different depicted simulated construction processes.

9. The principle of proximity compatibility: In the HSV framework's implementation in this case study, the models representing the object classes, "Cranes" and "Modules" are located on the same visualization screen to depict the simulated event of lifting and installing a module. Exceptions to this principle could happen when the framework has multiple visualization screens depicting various simulated processes taking places concurrently at different physical locations of the project.

10. The principle of multiple resources: The displaying of pathfinding mechanism findings followed this principle. The pathfinding mechanism implemented inside the visualization component found the shortest path for

each heavy lift crane mobilization event triggered by the simulation. The visualization depicts this shortest path using two formats; a graphical format and text format. This use of multiple formats for creating the visualization of these paths is a demonstration of the multiple resources principle.

D- Memory principles:

11. The principle of replacing memory with visual information: In the HSV framework's implementation in this case study, the shortest path parameters calculated throughout the project's life cycle simulation for the mobilization of crane events, are displayed to the decision-maker in a tabular text format, so he/she can retrieve any of this data at a later time of the simulation run. Also, the visualization screens in the HSV framework was not crowded with unnecessary or secondary information that is not related to the visualized simulated construction operation.

12. The principle of predictive aiding: The use of a game engine inside the framework helps in this matter due to its loose coupling to the simulation component and its ability to establish two-way communications with the simulation. This allows the decision-maker to visualize various construction scenarios. This will be discussed further in Chapter 3. This makes the decision-maker more proactive.

13. The visualizations inside the framework should be consistent: One of the strengths of this HSV framework is that it allows for more than one visualization screen to concurrently depict various construction operations. This was the case in this case study where two visualizations were working concurrently to depict the simulated operations taking place at the module yard and on site. Both visualizations screen designs should be consistent, with no major discrepancies between them. For instance, modules should be represented using the same 3D representations in the module yard and the site construction visualization screens.

This section discussed the means by which the visual analytics and display design principles discussed earlier in Chapter 2 aided in the conceptual design of the simulation visualization screens and simulation states translations into visualization behaviors. The full set of display design principles was not strictly utilized in the mapping of every simulation state into its corresponding visualization behavior; however each mapping was based on some of those principles.

4.4 Stochastic versus Deterministic Data Modeling of Cranes' Shortest Travelling Distance in a Dynamically Changing Site Layout

In this section, the HSV framework's PME output (cranes' shortest routes— travel distances— throughout this case study implementation) is modeled into probabilistic distribution functions. This is done to test if probabilistic distributions can used to stochastically estimate a resource's minimum travel duration given a certain site layout instead of the proposed HSV framework.

The output from implementing the HSV framework and its PME in this Ft. Saskatchewan site, (which contains the cranes' travel distances on feasible and least- expensive paths) was modeled into different distributions. Goodness of fit tests were used to assess the fitness of the theoretical distribution to the minimum travelling distance data at various significance levels.

The Cumulative Distribution Functions (CDF) for the fittest distributions will be shown together with the results of running the goodness of fit tests on those distributions. In light of that, the following sections will discuss the reasons for using the pathfinding mechanism rather than stochastic probabilistic distributions when calculating/estimating the shortest resource travelling distances in a dynamically changing site layout.

4.4.1 Used calculations and notes about Experiment

The theoretical and the empirical CDFs were compared at a range of relatively lower significance levels. The cranes' travel distances on shortest paths do not need to be tested at higher significance levels, as these distances are more of an approximation of the actual expected travel distance. The significance level is represented by the symbol gamma (α). The goodness of fit statistic values were compared to their critical statistic values at gammas equivalent to .2 and .1, corresponding to significant levels of 80% and 90%, respectively.

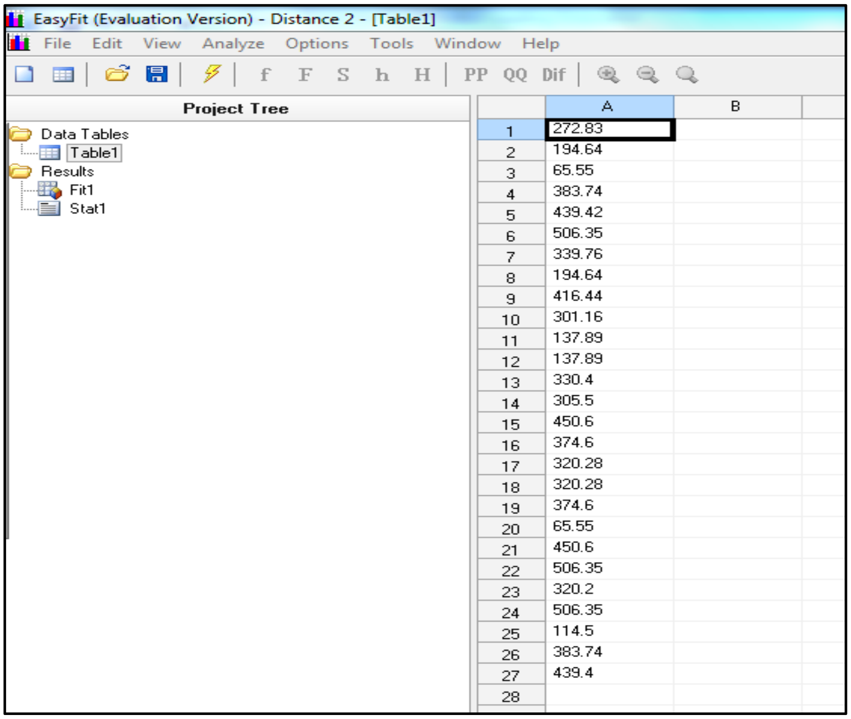
Three tests were used to assess the best CDF fit for modeling resource travel distance on site: the Visual test, Kolmogorov-Smirnov (K-S) test and Chi Squared test. The Visual test is a visual comparison between the travelled distance sample CDF curve and the theoretical curve of the specified distribution, also comparing the empirical data to the theoretical data using a P-P plot.

Once the statistic of each test is calculated for the travel distances data, they are compared to the critical values provided by the software based on the significance levels mentioned earlier. If the test statistic is greater than the critical values provided by the software at a given significance level, then the hypothesis that the travel distances data follow the distribution is rejected.

Next, the fittest distributions will be shown, together with their equations, CDF plots, and P-P plots. A discussion will follow on whether to use the stochastic estimate from these distributions or the deterministic values from the HSV framework's implemented pathfinding mechanism.

4.4.2 Best Fit Distributions: Evaluations and Analysis

Figure 4-19 shows a sample of the shortest cranes' travel distances which were calculated by the pathfinding mechanism after a complete simulation run of the project. This is the data to be fit to a probabilistic distribution.



	A	B
1	272.83	
2	194.64	
3	65.55	
4	383.74	
5	439.42	
6	506.35	
7	339.76	
8	194.64	
9	416.44	
10	301.16	
11	137.89	
12	137.89	
13	330.4	
14	305.5	
15	450.6	
16	374.6	
17	320.28	
18	320.28	
19	374.6	
20	65.55	
21	450.6	
22	506.35	
23	320.2	
24	506.35	
25	114.5	
26	383.74	
27	439.4	
28		

Figure 4-19 Shortest cranes' travel distance samples given by the pathfinding mechanism for a complete project simulation run

Based on the aforementioned goodness of fit tests, the distributions that fit best for this sample data will now be discussed.

4.4.2.1 Log-Pearson 3 Distribution

Figures 4-20 and 4-21 show the CDF and P-P plot for the Log-Pearson 3 distribution versus the sample data respectively.

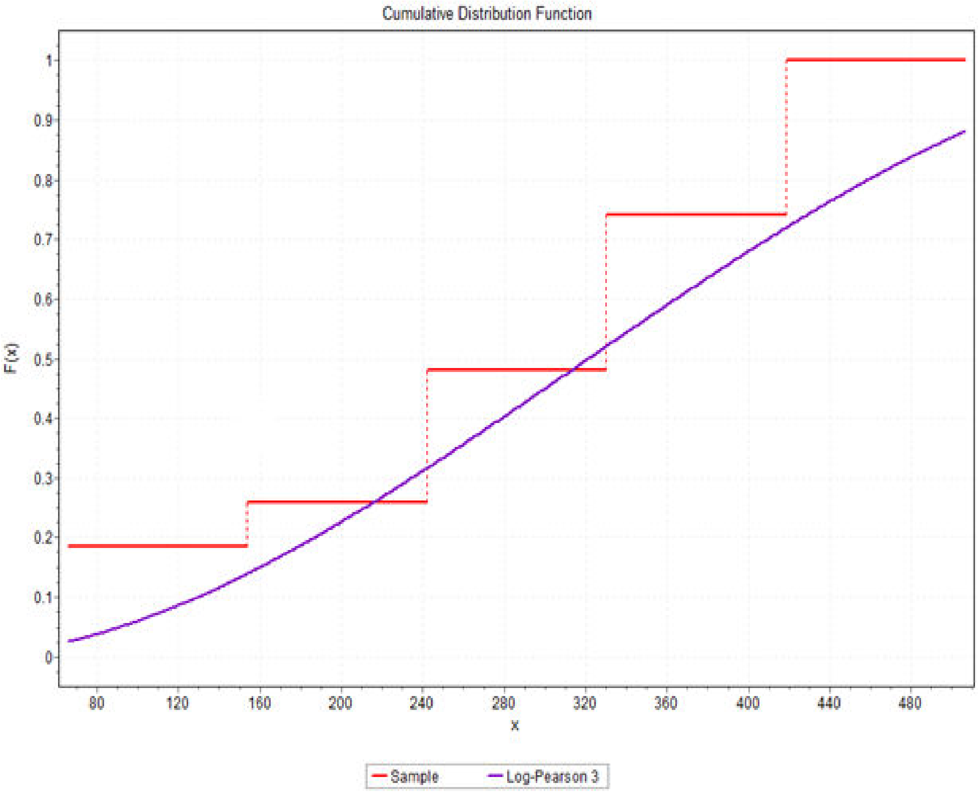


Figure 4-20 CDF for the Log-Pearson 3 distribution versus the sample data

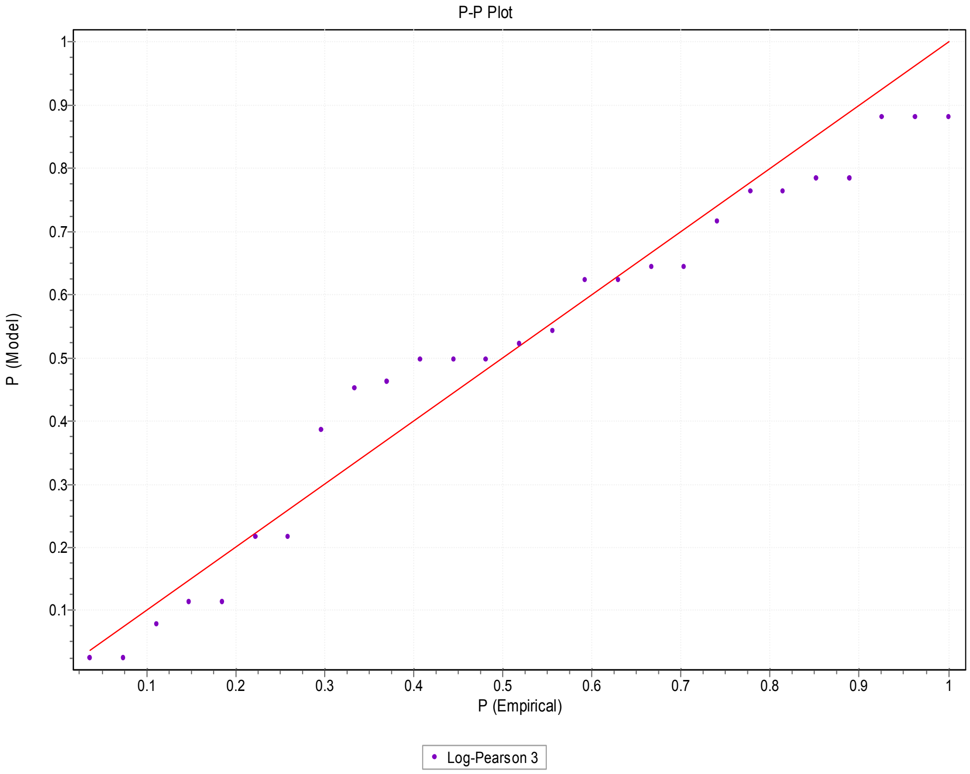


Figure 4-21 P-P plot for the theoretical Log-Pearson distribution CDF values versus their empirical CDF values

The CDF of this distribution is represented as,

$$F(x) = \frac{\Gamma_{(\ln(x)-\gamma)/\beta}(\alpha)}{\Gamma(\alpha)} \qquad \ldots\ldots (8)$$

Where, α is a continuous parameter where α>0

β is a continuous parameter such that $\beta \neq 0$

γ is a continuous parameter

The parameters for this sample distribution were as follows;

α = 2.09

β = -.4

γ = 6.4

As shown in Figure 4-20, the data sample is around the Log-Pearson 3 theoretical CDF distribution. Figure 4-21 (P-P plot) shows a clearer comparison: the 27 CDF value points against their theoretical CDF values. This plot should be approximately linear if the specified theoretical distribution is the correct model. As shown in Figure 4-21, the linearity is not clear, although according to the K-S and Chi-Squared statistics obtained, this was one of the best fitting distributions.

Table 4-3 shows the K-S and Chi-Squared test results for the Log-Pearson 3 distribution.

Kolmogorov-Smirnov		
Sample Size Statistic	27 0.15657	
α	0.2	0.1
Critical Value	0.2003	0.22898
Reject?	No	No
Chi-Squared		
Deg. of freedom Statistic	3 0.0594	
α	0.2	0.1
Critical Value	4.6416	6.2514
Reject?	No	No

Table 4-3 Goodness of fit values and the critical values at the corresponding significance levels for Log-Pearson 3 distribution

As shown in Table 4-3, both test statistics are less than their corresponding critical values at both significance levels. To the contrary of visual testing the test statistics showed that this distribution should not be rejected.

4.4.2.2 Generalized Extreme Value Distribution

Figures 4-22 and 4-23 show the Generalized Extreme Value Distribution versus the sample data CDF and P-P plot respectively.

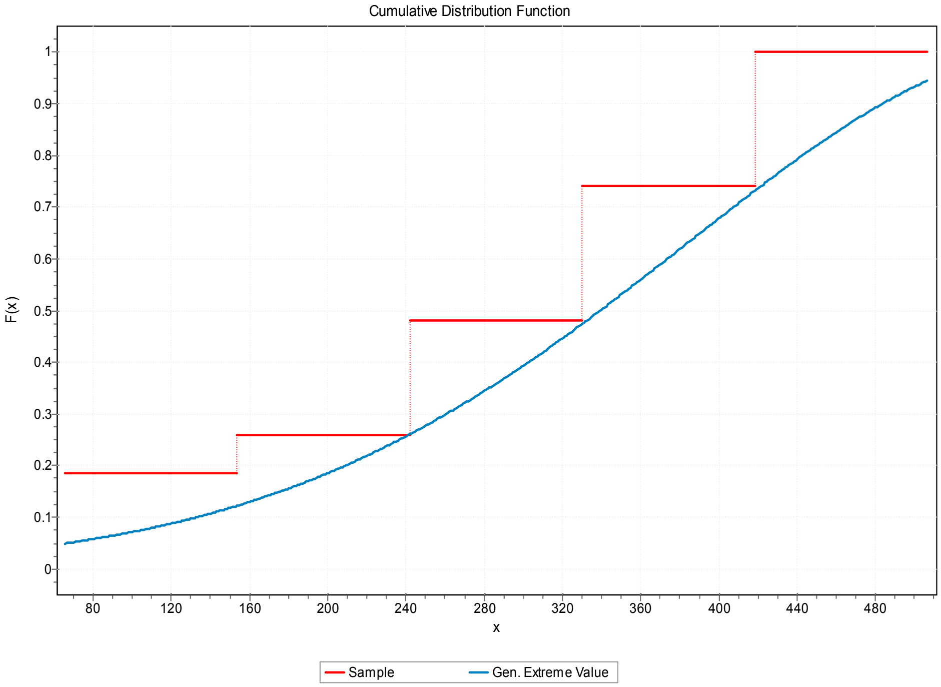


Figure 4-22 CDF for the Generalized Extreme Value Distribution versus the sample data

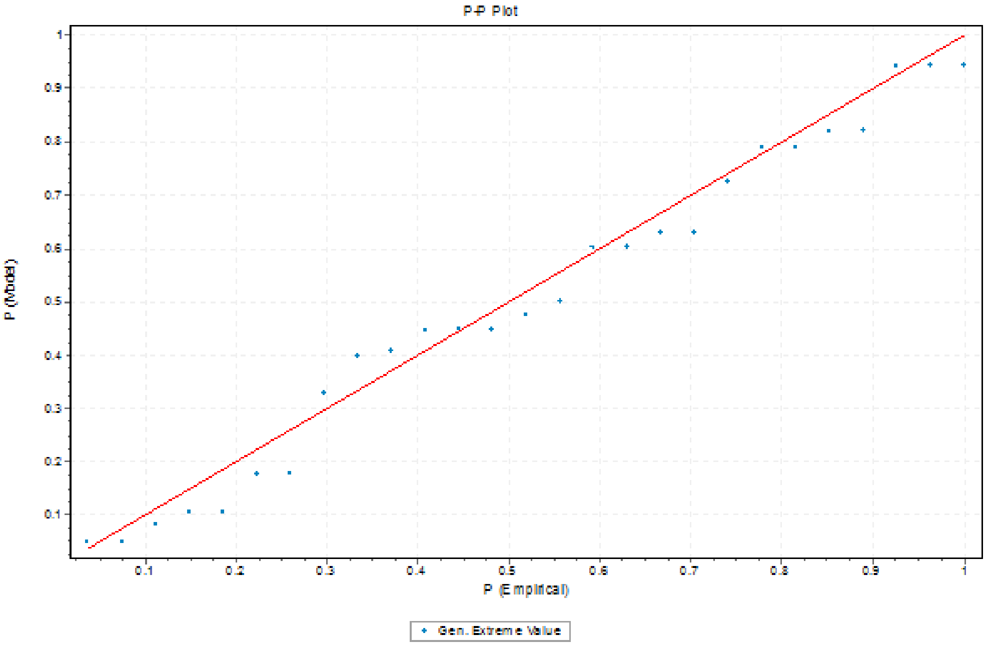


Figure 4-23 P-P plot for the theoretical Generalized Extreme Value Distribution CDF values versus their empirical CDF values

The CDF of this distribution is represented as,

$$F(x) = \begin{cases} \exp\left(-(1+kz)^{-1/k}\right) & k \neq 0 \\ \exp(-\exp(-z)) & k = 0 \end{cases} \qquad \text{...... (9)}$$

$$\text{where} \quad z \equiv \frac{x-\mu}{\sigma} \qquad \text{............. (10)}$$

Where, *K* is continuous shape parameter

μ is continuous shape parameter

σ is continuous shape parameter such that (*σ>0)*

The parameters for this sample distribution were as follows:

K = -.54

μ = 289.8

σ = 148.9

Figure 4-23 (P-P plot) shows, yet again, a clearer comparison of the 27 CDF value points against their theoretical CDF values. As was the case with the P-P diagram of the Log-Pearson 3 Distribution, the linearity is not clear; however there is more linearity in this case than in the Log-Pearson 3 Distribution. Also the K-S statistic of the Generalized Extreme Value Distribution (shown in Table 4-4) is lower than that of the Log-Pearson 3 (shown in Table 4-3).

Kolmogorov-Smirnov		
Sample Size Statistic	27 0.10033	
α	0.2	0.1
Critical Value	0.2003	0.22898
Reject?	No	No
Chi-Squared		
Deg. of freedom Statistic	3 0.47627	
α	0.2	0.1
Critical Value	4.6416	6.2514
Reject?	No	No

Table 4-4 Goodness of fit values and the critical values at the corresponding significance levels for Generalized Extreme Value Distribution

To summarize, the two best fitting distributions from a number of distributions were discussed above. Among the distributions, the Generalized Extreme Value Distribution seems to fit best, yet the fit is not perfect. The next section will discuss the validity of using this distribution to estimate cranes' shortest paths in a dynamically changing site layout.

4.4.3 Validity of Using Stochastic Distribution to Estimate Heavy Lift Mobile Resources' Shortest Paths

It is important to note that estimating the shortest resource paths from the stochastic Generalized Extreme Value Distribution can be misleading for a number of reasons:

- Although the K-S test statistic was satisfactory, the visual assessment of the distribution fit was not. Overall, one can still be hesitant to model heavy lift mobile resources' estimated shortest paths travel distance using this stochastic distribution.
- The shortest distance is estimated here, not the average travelling distance of mobile resources onsite. If the travel distance is estimated randomly from this stochastic distribution, there is a high probability that the result will exceed the

distance of the shortest path. The HSV framework uses a pathfinding mechanism to estimate the shortest distance for the least- expensive path rather than an average distance that is modeled based on past experience and observations.

- Bearing in mind that no construction sites have identical layouts, a stochastic distribution (Generalized Extreme Value Distribution) that is sampled based on observations from one site will not be indicative of mobile resources' shortest paths in a different site. Therefore, it is misleading to use this stochastic distribution in estimating shortest paths in other sites.
- The fitted distributions do not consider the time effect at all. This means that as the project time progresses, site spaces become more congested and occupied. This congestion factor has to be taken in consideration (modeled) when estimating the heavy lift mobile resources (heavy lift cranes) mobilization times at later stages of a project.
- Estimating heavy lift mobile resources shortest path distances from stochastic distributions is "static" in nature, meaning that such distributions do not take the dynamic changes in a site layout into consideration. The pathfinding mechanism inside the framework considers the continuous changes in a site layout and arrives at a deterministic value. The deterministic values calculated by the pathfinding mechanism inside the HSV framework take into consideration the safety factor. That is because the calculated resources' shortest paths must also be obstacle-free.

Based on the above arguments, none of the tested distributions can perfectly replace the detailed approach using the HSV framework to find the heavy lift cranes mobilization durations. The best way to plan for minimizing and finding resource mobilization durations in a dynamically changing site layout is through the combination of simulation driven visualization and search algorithms (HSV framework), rather than the use of stochastic distributions. Consequently, the pathfinding mechanism was implemented inside the HSV framework to achieve this target as described earlier.

Chapter 5: Conclusion and Further Research

5.1 Conclusion

This research presented a new way of addressing the simulation modeling limitation when it comes to spatial data representation. It presented a new HSV framework to address this problem. The framework used the distributed simulation HLA standards. It built on the inherent strengths of both simulation and visualization that resulted in an automated and novel way of solving heavy lift resources pathfinding problem in a dynamic site layout based on modeling the spatial changes in this site. The HSV framework showed good potential as an interactive dynamic site layout planning tool in terms of modeling dynamic changes in site layout and modeling changes in sites' geometries. Also it was a useful tool in solving the heavy lift resources pathfinding problem.

The framework's ability to depict changes in construction site's spaces geometry was utilized in the construction industry to achieve the following:

- Models both: changes in site space and the dynamic changes in site layout.
- Acts as a planning tool for calculating the heavy lift resources shortest safe paths' distances and expected durations for those resources mobilizations on those paths.

Decision-makers can use the framework's pathfinding mechanism outputs to make educated decisions based on the following:

- Finding shortest resource paths onsite.
- Modeling future changes in site space and layout.
- Minimizing the total heavy lift resources' mobilization events duration.
- Testing whether the already existing heavy lift resources' paths are the shortest, obstacle-free ones.
- Clearing the predefined paths for heavy lift resources mobilization throughout the project lifecycle.

In order for the HSV framework to achieve the above mentioned goals, it had to address some limitations that exist in the current state of the art construction SDV mechanisms.

The HSV framework has the following advantages over other construction SDV frameworks:

- Two-way communication between the simulation and visualization components of the framework allowing the information to flow in both directions between the components. This enhances the use of the framework in the decision making process.
- Loose coupling between the simulation and visualization components. This enhances the re-usability of the framework and its application to various construction processes and projects.
- The HSV framework's underlying distributed simulation architecture allowed for multiple concurrent construction process simulations and visualizations to be used for the same project. This allows the decision-makers to concentrate on various aspects and processes of the project. It also allows multiple decision-makers who physically exist in various locations to concurrently experiment with various simulation scenarios and/or site layout scenarios inside the framework.
- The HSV framework is robust in terms of its ability to build on the strengths of both its visualization and DES components. There was no need to make any compromises on the strengths of any of the simulation or visualization components to achieve interoperability and concurrent two-way simulation visualization.

This thesis started by a critical review of the construction literature related to the objectives and tools used in this research. The Visual analytics and display design principles used in the conceptual design of the framework's visualization component in the Ft. Saskatchewan case study (Chapter 4) were also discussed. Chapter 3 discussed the complete technical developments of the framework with its PME. The chapter also explained how the framework with its PME will be utilized in modeling a construction site's spatial changes and heavy lift mobile resources' tempo-spatial planning. The mesh generation mechanism and A* algorithm implemented inside the framework to perform resources' tempo-spatial planning were also explained in this chapter. In Chapter 4, the HSV framework with its full technical developments and pathfinding mechanism was applied to a real construction project case study (Ft. Saskatchewan oil upgrader site) to judge and demonstrate its ability to depict/model future changes in site spaces and perform heavy lift cranes tempo-spatial planning. This case study implementation showed that the HSV framework can be utilized to validate the simulation's logic and chosen construction sequence. The conceptual design process of the visualization component of the HSV framework that was utilized in this case study, according to visual analytics and display design principles, was also explained

5.2 Research Contributions

The proposed HSV framework will contribute towards the construction industry through integrating site space management and resources' pathfinding capabilities. The framework makes use of the SDV ability to model future changes in site space and layout. The framework makes use of this integration to act as a planning tool for calculating the heavy lift resources shortest safe paths' distances and expected durations for those resources mobilizations on those paths.

5.3 Further Research

The limitations mentioned earlier in Section 3.4.3 should be addressed, for example; the safety buffer zone should be adjusted based on the movable resource's print shape. Also, the HSV framework's 2-way interaction can be further enhanced to act like a site management tool in a game-like environment. Also, the resource path problems can be treated as "Robot Navigation" problems, where the resource has an infinite rather than a discrete set of routes to move in a dynamic site layout. The framework can also be applied to other construction problems other than resources' mobilizations planning. A graphical user interface can be added to the framework to allow the decision-makers to customize the framework visualization in order to adapt to the construction problem they want to solve. This interface would make the framework more customizable to solve more space modeling problems in construction projects/sites of various categories.

Bibliography

Al-Hussein M., Niaz M., Yu H. and Kim H. (2006). Integrating 3D visualization and simulation for tower crane operations on construction sites, *Journal of Automation in Construction,* Vol. 15, No.5, September 2006.

AlBahnassi, H., Hammad, A., and Zhang, C. (2009). Accurate heavy equipment motion planning considering local and global constraints, *Proceedings of Annual Conference - Canadian Society for Civil Engineering,* St. Johns, NL, Canada, May 2009.

Blender Foundation. (2010). Help and Education [Online] Available at: http://www.blender.org/education-help/.

Boff K., Kaufman L. and Thomas J. (1986). Handbook of perception and human performance, Wiley, NY, USA.

Card S., Mackinlay J. and Shneiderman B. (1999). Readings in information visualization: Using vision to think, Morgan Kuffman Publishers Inc., San Francisco, CA, USA.

Cawthon N., and Moore A. (2007). The effect of aesthetic on the usability of data visualization, *Proceedings of the International Conference on Information Visualization*, Zurich, Switzerland, July 2007.

Dawood N. and Marasini R. (2003). Visualization of stockyard layout simulator "Simstock": A case study in precast concrete products industry, *Journal of Automation in Construction,* Vol. 12, No.2, March 2003.

Dawood N., Scott D., Sirprasert E., and Mallasi Z. (2005). The virtual construction site (VIRCON) tools: An industrial evaluation, *Journal of Information Technology in Construction,* Vol.10, 2005.

El-Rayes K. and Said H. (2009). Dynamic site layout planning using approximate dynamic programming, *Journal of Computing in Civil Engineering*, Vol. 23, No.2, 2009.

Elbeltagi E., Hegazy T. and Eldosouky A. (2004). Dynamic site layout of construction temporary facilities considering safety, *Journal of Construction Engineering and Management*, Vol.130, No.4, July-August 2004.

ElNimr A. and Mohamed Y. (2010). A simulation driven visualization framework for construction operations: Development and operation, *Proceedings of the Construction Research Congress*, Banff, AB, Canada, May 2010.

ElNimr A. and Mohamed Y. (2011). Application of gaming engines in simulation driven visualization of construction operations, *Journal of Information Technology in Construction*, Vol.16, January 2011.

Gominuka, T. and Sadeghpour, F. (2008). A framework for activity-based identification of space requirements for construction equipment, *Proceedings of the Annual*

Conference of the Canadian Society for Civil Engineering 2008 "Partnership for Innovation", Quebec City, QC, Canada, June 2008.

Hajjar D. and Abourizk S. (1999). Simphony: an environment for building special purpose construction simulation tools, *Proceedings of the Winter Simulation Conference* , Phoenix, USA, December 1999.

Halpin D. and Woodhead R. (1976). Design of construction and process operations, John Wiley and Sons, New York, N.Y., USA.

Hessom D., and Mahdjoubi L. (2004). Trends of 4D CAD applications for construction planning, *Journal of Construction Management and Economics,* Vol.22, No.2, February 2004.

Huang R. and Halpin D. (1994). Simulation of cable stayed bridges using DISCO, *Proceedings of the Winter Simulation conference*, Buena Vista, USA, December 1994.

Ioannu P. and Martinez J. (1996). Animation of complex construction simulation models, *Proceedings of the Congress of Computer in Civil Engineering*, New York, USA, 1996.

Kalk A. and Douglas S. (1980). *Insight: Interactive simulation of construction operations using graphical techniques, Technical Report No. 283*, Construction Institution, Department of Civil Engineering, Stanford Univ., Stanford, California, USA.

Kamat V. and Matrinez J. (2001). Visualizing simulated construction operations in 3D, *Journal of Computing in Civil Engineering*, Vol. 15, No.4, October 2001.

Kamat V. and Martinez J. (2002). Comparison of simulation-driven construction operations visualization and 4D CAD, *Proceedings of the Winter Simulation Conference*, San Diego, CA, USA, December 2002.

Kamat V. and Matrinez J. (2004). General Purpose 3D-Animation with VITASCOPE, *Proceedings of the Winter Simulation Conference,* Washington, DC, USA, December 2004.

Kamat V. and Matrinez J. (2008). Software mechanisms for extensible and scalable 3D visualization of construction operations, *Journal of Advances in Engineering Software*, Vol. 39, No.8, August 2008.

Ma Z., Shen Q. and Zhang J. (2005). Application of 4D for dynamic site layout and management of construction projects, *Journal of Automation in Construction,* Vol. 14, No.3, June 2005.

Mckinney K., and Fischer M. (1998). Generating, evaluating and visualizing construction schedules with CAD tools, *Journal of Automation in Construction,* Vol.7, No.6, September 1998.

Norman D. (1988). *The psychology of everyday things*, Harper &Row, NY, USA.

Oloufa A. (1993). Modelling and simulation of construction operations, *Journal of Automation in Construction*, Vol. 1, No.4, March 1993.

Oloufa A.A. and Crandall K.C. (1992). Feedback mechanism for operational simulation, *Journal of Computing in Civil Engineering*, Vol. 6, No. 2 , April 1992.

Parasuraman R., Davies R. and Beatty J. (1984). *Varieties of attention*, Academic Press, NY ,USA.

Paulson B.C., Douglas S.A., Kalk A., Touran A. and Victor G. (1983). Simulation and analysis of construction operations, *Journal of Technical Topics in Civil Engineering*, Vol. 109, No. 2 , August 1983.

Rekapalli P. and Martinez J. (2009). Runtime user interaction in concurrent simulation-animations of construction operations, *Journal of Computing in Civil Engieering,* Vol.23, No.6, December 2009.

Rekapalli P. and Martinez J. (2011). Runtime user interaction in concurrent simulation-animations of construction operations, *Journal of Construction Engineering and Management,* Vol.137, No.3, March 2011.

Rohrer M. and McGregor I. (2002). Simulating reality using Auto-Mod, *Proceedings of the Winter Simulation Conference*, San Diego, USA, December 2002.

Rosceo S. (1968). Airborne displays for flight and navigation. *Human Factors,* Vol.10, No.4, August 1968.

Russell A. and Udaipurawla A. (2004). Using multiple views to model construction, *Proceedings of CIB World Congress Building,* Toronto, ON, Canada, May 2004.

Russell A., Chiu C. and Korde T. (2009). Visual representation of construction management data, *Journal of Automation in Construction,* Vol.18, No.8, December 2009.

Russell S. and Norvig P. *Artificial intelligence a modern approach, 2nd edition,* Prentice-Hall, Upper Saddle River, N.J., USA.

Sadeghpour F., Moselhi O. and Alkass S. (2006). Computer aided site layout planning, *Journal of Construction Engineering and Management*, Vol. 132, No.2.

Sadeghpour, F., Moselhi, O. and Alkass, S. (2005). Modeling the time factor in site planning, *Proceedings of the Annual Conference of Canadian Society for Civil Engineering*, Toronto, ON, Canada, June 2005.

Schmid C. (1978). The role of standards in graphic presentation, *Proceedings of the Annual Meeting of the American Statistical Association,* USA, 1978.

Shannon R.E. (1975). *System Simulation: The Art and Science*, Prentice-Hall, Englewood Cliffs, N.J., USA.

Sirprasert E. and Dawood N. (2003). Multi-constraint information management and visualization for collaborative planning and control in construction, *Journal of Information Technology in Construction,* Vol.8, 2003.

Song K., Pollalis S. and Pena-Mora F. (2005). Project dashboard: concurrent visual representation method of project metrics on 3D building models, *Proceedings of ASCE International Conference on Computing in Civil Engineering,* Reston, VA, USA, 2005.

Songer A., Hays B. and North C. (2004). Multidimensional visualization of project control data, *Journal of Construction and Innovation,* Vol.4 , No.3, 2004.

Thomas J. and Cook K. (2005). *Illuminating the path: The research and development agenda for visual analytics*, IEEE Computer Society Press, Los Alamitos, CA, USA.

Truevision3D. (2010). 3D Engine and Game Development SDK by Truevision3D [Online] Available at: http://www.truevision3d.com/.

Wickens C. and Hollands J. (2000). *Engineering psychology and human performance, 3rd edition*, Prentice-Hall, Upper Saddle River, NJ, USA.

Wickens C., Lee J., Liu Y. and Becker S. (2004). *An introduction to human factors engineering, 2nd edition*, Prentice-Hall, Upper Saddle River, NJ, USA.

Xu J. and Abourizk S. (1999). Product-based model representation for integrating 3D CAD with computer simulation, *Proceedings of the Winter Simulation Conference*, Phoenix, USA, December 1999.

Zeb J., Chiu C. and Russell A. (2008). Designing a construction data visualization environment, *Proceedings of the 1st Forum on Construction Innovation,* Montreal, Quebec, Canada, 2008.

Zhang J., Anson M. and Wang Q. (2000). A new 4D management approach to construction planning and site space utilization, *Proceedings of the International Conference on Computing in Civil and Building Engineering,* August 2000.

Zhang X., Bakis. N.,Lukins T., Ibrahim Y., Wu S., Kagioglou M., Aouad G., Kaka A. and Trucco E. (2009). Automating progress measurement of construction projects, *Journal of Automation in Construction*, Vol. 18, No. 3, 2009.

Zhou F., Abourizk S. and Al-Battaineh H. (2009). Optimization of Construction site layout using a hybrid simulation-based system, *Simulation Modeling Practice and Theory*, Vol.17, No.2, February 2009.

Appendix I: Visualization of Construction Projects Bidding Game Simulation

A first sample small implementation was carried out in order to make sure that the proposed framework architecture is interoperating properly and integration is enabled using this architecture. The implementation was aimed at providing some visual output to a bidding game developed as a training tool for construction students to learn how the bidding process in the construction industry takes place. Players take the roles of general contractors, who bid on projects published by a Market Simulator Federate. The players can compete against each other and/or against virtual players driven by computer algorithms. The bidding game federation is composed of the federates shown in Figure 3-5; Table 3-1 explains the functions of the federates.

The objective of the visualization component in this federation is to provide visual clues to the locations and natures of existing and upcoming projects in the game. The Visualization Federate was developed using TureVision 3D® (a commercial game engine) as the graphics engine. This engine provides a set of libraries to manipulate 3D graphics and an advanced Application Programming Interface (API) for Microsoft .NET. This API allows developers to load 3D models and manipulate them without the need for extensive programming experience. Although this implementation does not map detailed resource interaction of construction operations, it demonstrates the use of a gaming engine to provide some visual output to a bidding game

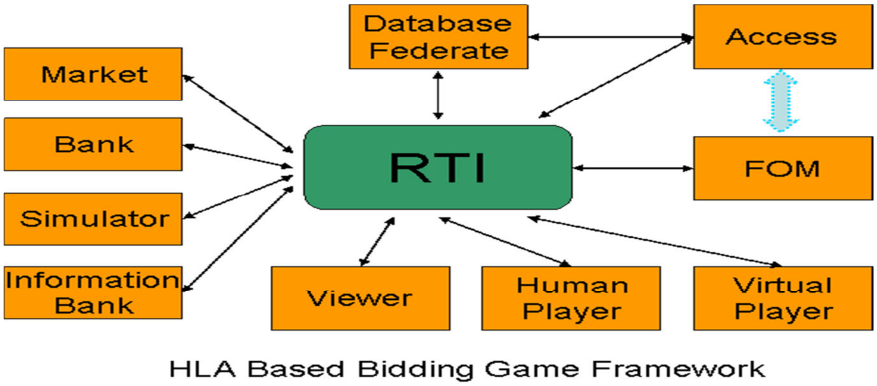


Figure 1 Federates forming the Bidding Game Federation

Federate	Function
Market	Simulates a construction market and generates projects with attributes for bidding. Generates subcontractors bids
Bank	Simulates the bank for human players, giving them loans, lines of credit, etc.
Simulator	1) Advances game time (simulation time); 2) Collects bids; 3) Awards bids; 4) Deposits money to each player's account at the end of each period, according to the pre-defined payment schedule of each project type; 5) Tracks actual progress of each project. Compares the actual schedule and cost to what was planned in the bids and publish that information.
Info bank	1) Records the historical performance of each Subcontractor; 2) Records projects from past or current periods. 3) Publishes project information in the next period for the players (general contractors)
Viewer Federate based on TureVision Gaming Engine	Displays project attributes and locations to inform human players and the administrator of the projects available for bidding during the simulation run
Human player	1) Acquires loan; 2) Submits bid; 3) Selects subcontractors; 4) Requests information from information bank.
Virtual player	Same as the human player
Access Database	Stores a standard project warehouse, and defines the properties of each project category, including the area of building, the actual cost of each period for each project type, the unit cost for each sub-trade, and so on.

Table 1 The function of each federate in the Bidding Game federation

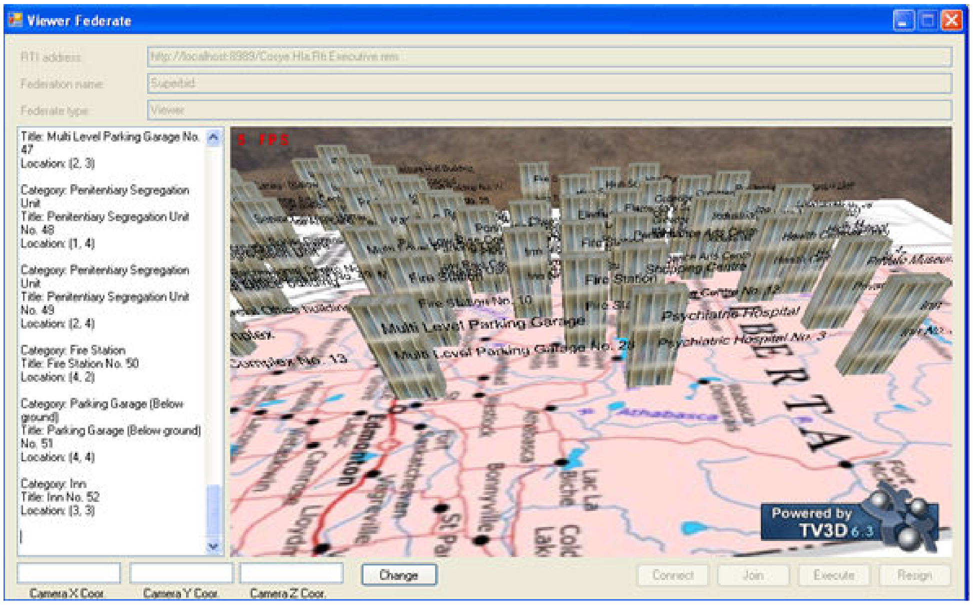
The focus of this case study is only on the Viewer Federate and how it will interoperate with the Discrete Event Simulation (DES) component in the framework. A 3D display surface is developed through the TureVision engine and embedded inside the Viewer Federate window, which connects to the RTI once simulation starts. The development is all carried out using Visual Basic .NET code. The Viewer Federate in this implementation acts as a listener to interactions and attribute updates produced by other federates in the federation. Once the Viewer Federate is connected to the simulation, and the simulation starts, the Market Federate starts to generate projects and their attributes at different times during the simulation.

The Viewer Federate listens (subscribes) to a number of attributes of a project. These attributes are:

- Project location: Each project is generated at a certain location defined in the form of (X, Y) coordinates. This (X, Y) location is read by the Viewer Federate and scaled to fit the dimensions of a map of the province of Alberta (the province for which the market is simulated). Once the project is published in the federation, the Viewer Federate displays its location on the map using a generic 3D model of a building.

- Project title and category: The Viewer Federate displays the title and category for each project. These attributes are displayed using a 3D font in front of each project on the display surface.

Human players use the visualization to pick projects for bidding and to know when projects are generated during the simulation run. Figure 3-6 shows a screen shot of the Viewer Federate during the simulation while Figure 3-7 shows the logic behind the code used for utilizing the gaming engine in the development of the Viewer Federate.



Viewer Federate

RTI address: http://localhost:8989/Cosye.Hla.Rti.Executive.rem

Federation name: Superbid

Federate type: Viewer

Title: Multi Level Parking Garage No. 47
Location: (2, 3)

Category: Penitentiary Segregation Unit
Title: Penitentiary Segregation Unit No. 48
Location: (1, 4)

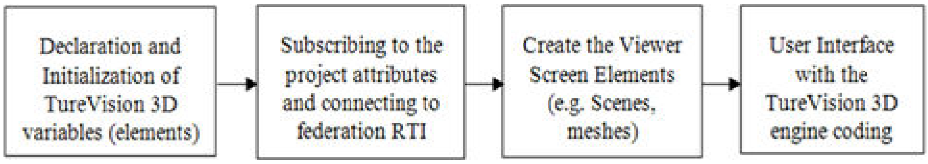
Category: Penitentiary Segregation Unit
Title: Penitentiary Segregation Unit No. 49
Location: (2, 4)

Category: Fire Station
Title: Fire Station No. 50
Location: (4, 2)

Category: Parking Garage (Below ground)
Title: Parking Garage (Below ground) No. 51
Location: (4, 4)

Category: Inn
Title: Inn No. 52
Location: (3, 3)

Figure 2Viewer Federate showing projects, their locations and project information during the simulation



Declaration and Initialization of TureVision 3D variables (elements)
Subscribing to the project attributes and connecting to federation RTI
Create the Viewer Screen Elements (e.g. Scenes, meshes)
User Interface with the TureVision 3D engine coding

Figure 3 TureVision 3D engine coding logic inside the Viewer Federate

The gaming engine pseudo-code shown in Figure 3-7 starts with the declaration of all the TureVision 3D elements; the various game engine components are declared here, for example, the TV Engine itself, scenes, meshes, cameras and the scene atmosphere. The code is then set to subscribe to the project's attributes mentioned earlier and read the changes in these attributes' values by connecting to the RTI. This subscription part includes the code to display projects based on their actual locations on the viewer screen. Then the code creates the various screen elements which were set in the declaration phase. These various elements (including meshes with their various textures representing different projects) are displayed to the players based on the simulation scenario. The last portion of the code sets the user interface, for example the user controlling the viewer screen through mouse movements or changing the scene's camera.

The use of a game engine in this implementation allowed for the development of the Viewer Federate in a relatively short time. It also allowed for integrating naturally with other components of the federation (i.e. other federates and the RTI) through the VB .NET API. However, it still required some programming knowledge for customizing the behaviors of the 3D objects on the display surface. Despite the advanced services provided by the engine, this programming knowledge grows significantly with the complexity of the behaviors required for the 3D objects.

This first implementation using the TureVision 3D engine enabled seamless integration with the HLA framework as both are .NET based. However, generating visual behaviors in response to simulation interactions using code required significant programming background despite the high level functionality that the engine provides. In addition, the engine provides functions to manipulate an existing set of objects but does not help much with the creation (modeling) of the 3D world, which is not a trivial task.

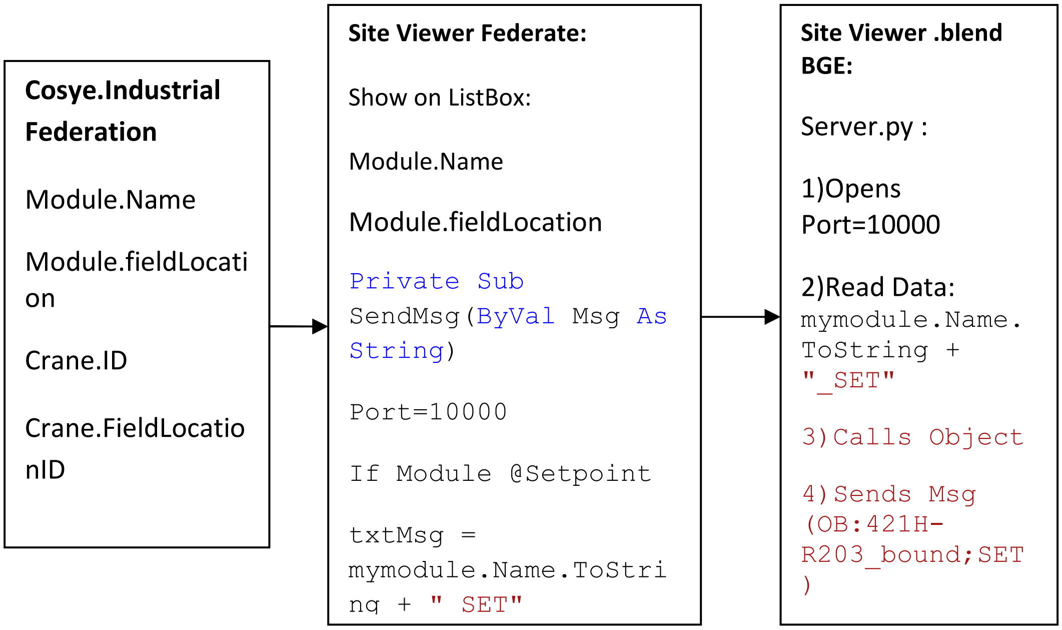
As a result of this experiment, a search for a more effective and flexible way to provide visual behaviors and interaction with 3D objects took place. The search resulted in the utilization of another gaming engine; Blender Game Engine (BGE) inside the visualization component of the framework applied in the main case study (Chapter 4). Now that the HSV framework interoperability succeeded in this first implementation, it was time to add the generic Pathfinding Mechanism Extension (PME) to the HSV framework.

Appendix II: Site Visualization Manual

The development of the site visualization federate and the industrial federation was done using Cosye framework which is a C# framework that allows developers to develop interoperable simulations using HLA standards. Cosy was developed by programming team at the University of Alberta's Hole School of Construction Engineering and it can be downloaded from http://irc.construction.ualberta.ca/cosye. The site and yard interactive visualization screens were developed through Blender©, which is an open source 3D engine with a gaming engine known as Blender Game Engine (BGE). Blender can be downloaded from this website: www.blender.org .

This manual is to further explain the technical details of the site visualization federate and site visualization Blender Game engine (BGE) important developments. The manual does not include all the developments' details; however it includes enough to put the reader on the right track to follow through this development for future simulation visualization developments using the HSV framework's components.

The way the logic is designed is such that a certain object inside the scene of the game acts as the Server to receive the message from the site visualization federate. The object is called: Federate HUB and it is the server which has the Server.py on its controller. While the Client in this case is the Siteviewer.vb (Site Viewer Federate). This Client (Site Viewer Federate) which sends reflections of Module.Name, Module.Fieldlocation , among other FOM resources object classes' attributes. The way it goes is as follows:



Cosye.Industrial Federation

Module.Name

Module.fieldLocation

Crane.ID

Crane.FieldLocationID

Site Viewer Federate:

Show on ListBox:

Module.Name

Module.fieldLocation

```
Private Sub SendMsg(ByVal Msg As String)

Port=10000

If Module @Setpoint

txtMsg = mymodule.Name.ToString + " SET"
```

Site Viewer .blend BGE:

Server.py :

1)Opens Port=10000

2)Read Data:
```
mymodule.Name.ToString + "_SET"
```

```
3)Calls Object

4)Sends Msg (OB:421H-R203_bound;SET)
```

Figure 1 The logical flow of one way of the HSV framework pipeline

1st: The Cosye Industrial Federation

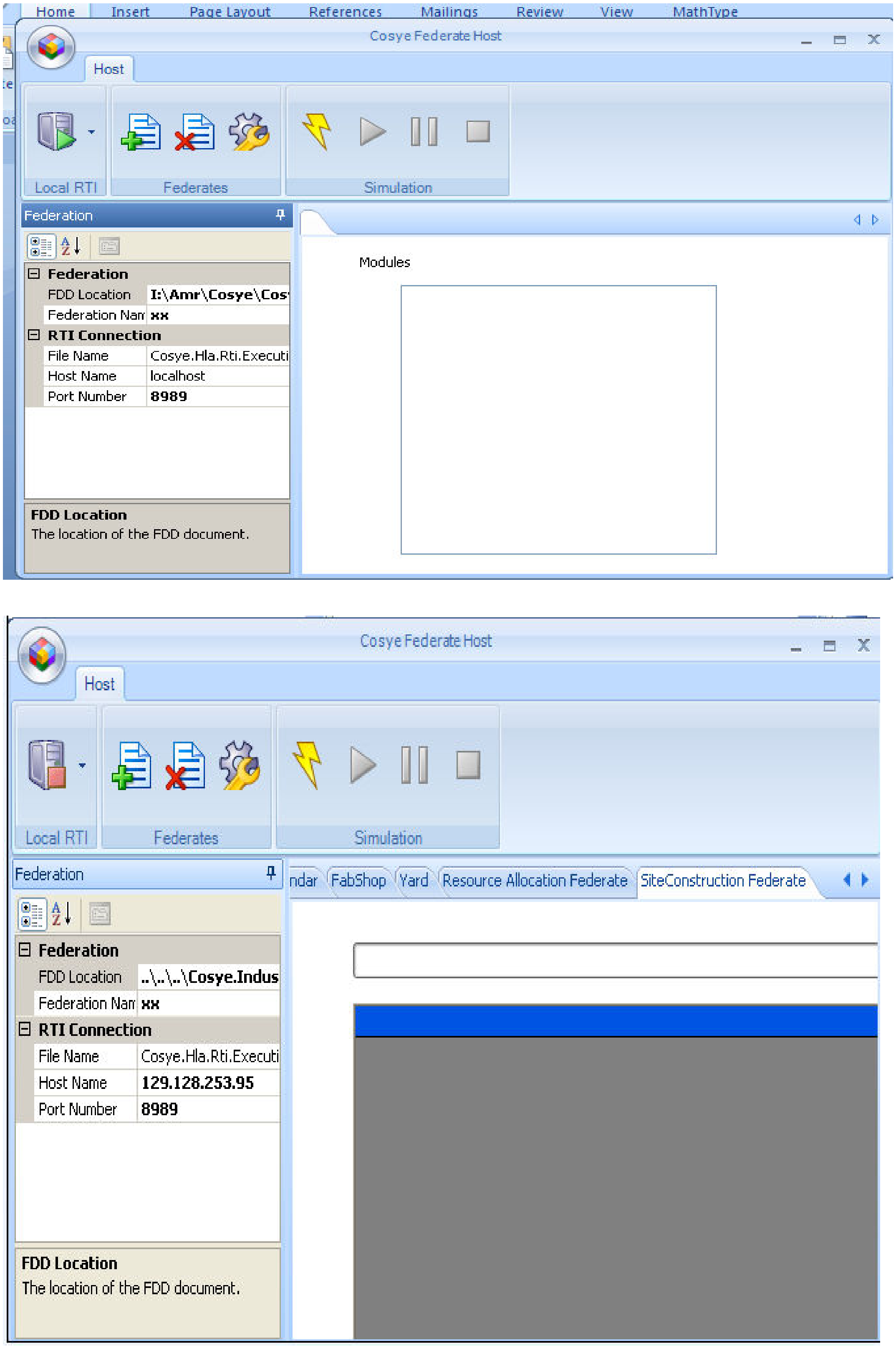
It has to be designed to publish whatever attributes that you want your client federate (Site Viewer in this case) to read and reflect to the BGE. In this case 4 attributes are read:

1. Module.Name
2. Module.FieldLocation
3. Crane.ID
4. Crane.FieldLocationID

The Federate Host which has Cosye.Industrial Federation has to be set on the following settings:

- IP of RTI : The computer which has the Site Viewer federate on it
- Federation Name: The Cosye.Industrial and Site Viewer Federates should have the same name in the "Federation name" field.
- Port Number: The client (Site Viewer Fed) and Cosye Industrial federation should read from the same port on the Fed Host form.

Figure 2 shows snapshots of federate host for both the Site Viewer Client Federate and the Cosye.Insustrial Federate:



Federation

FDD Location I:\Amr\Cosye\Cos

Federation Nam xx

RTI Connection

File Name Cosye.Hla.Rti.Executi

Host Name localhost

Port Number 8989

FDD Location

The location of the FDD document.

Federation

FDD Location ..\..\..\Cosye.Indus

Federation Nam xx

RTI Connection

File Name Cosye.Hla.Rti.Executi

Host Name 129.128.253.95

Port Number 8989

FDD Location

The location of the FDD document.

Figure 2 the Site Viewer Federate (Client) and the Cosye.Industrial Federation Federate Hosts

2nd: The Site Viewer Federate-Client

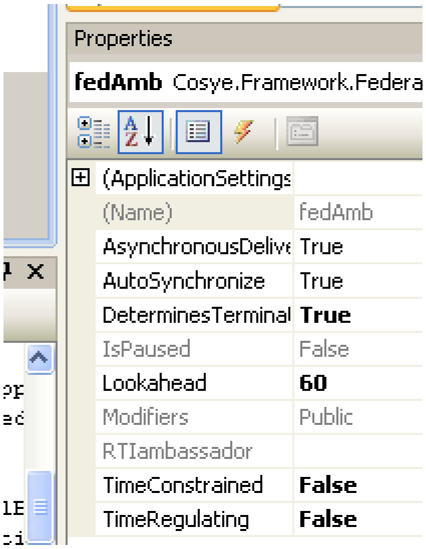
Now we are sure that the source simulation on the Cosye.Industrial. SiteConstruction is publishing both attributes the FieldLocation and Name; we need the Site Viewer Federate to receive reflections from the Industrial Federation (which may run on same or different terminal with a different IP address). The receiving of reflections here will be through the COSYE.RTI not through messages. The Site.viewer is only sending message to the BGE's Federation.HUB object, which acts as a server to the site.viewer client. This is very important to understand, that federates contacts are through the RTI on the same terminal as the Site.Viewer client.

Two questions that need to be answered when developing a viewer federate in this framework:

How do we connect to the federation portion through RTI?

The site viewer is built inside a federate host, by simply following the federate development example on: http://irc.construction.ualberta.ca/cosye. However, instead of inheriting a federate form with create; go for a "Controlled federate form" which comes with the downloadable Cosye package.

The viewer federate is not time regulating or constrained .See Figure 3 below.



Properties

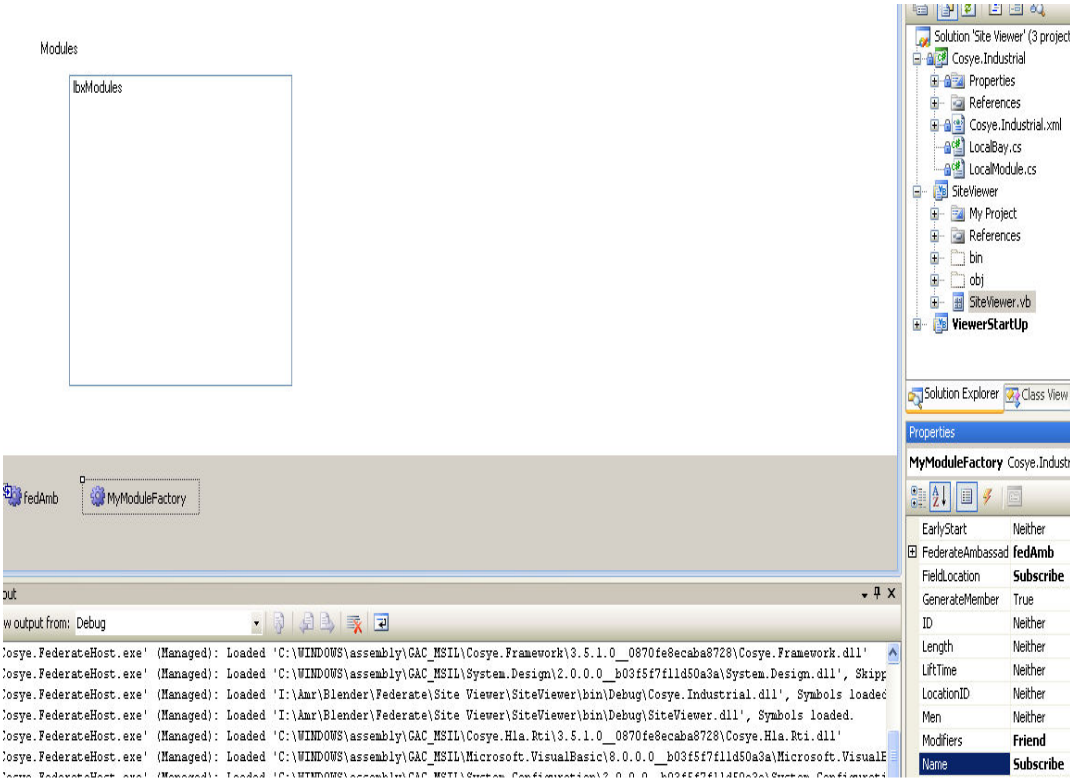
fedAmb Cosye.Framework.Federa

⊞ (ApplicationSettings	
(Name)	fedAmb
AsynchronousDelive	True
AutoSynchronize	True
DeterminesTerminal	**True**
IsPaused	False
Lookahead	**60**
Modifiers	Public
RTIambassador	
TimeConstrained	**False**
TimeRegulating	**False**

Figure 3 Site Viewer Federate Time Management Implementation

Now, make sure to add the list box (LbxModules in this case) which is generically there to display the messages coming from the source simulation federate (Cosye.Industrial

Federation in this case). The listbox is just there to show what the Viewer Federate receive from the federation, hence Figure 4 .Now once the list box is added , start adding the factories for the simulation objects that the site viewer will be subscribing to (in our case Module Factories –MyModuleFactory- and subscribe to the object's attributes that you will be receiving from the source simulation (see figure 4).Of course, a Factory needs to be added for each Object that the Viewer subscribes to its attributes (receives reflections from the Source Simulation Federation-RTI).



```
Cosye.FederateHost.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Cosye.Framework\3.5.1.0__0870fe8ecaba8728\Cosye.Framework.dll'
Cosye.FederateHost.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Design\2.0.0.0__b03f5f7f11d50a3a\System.Design.dll', Skipp
Cosye.FederateHost.exe' (Managed): Loaded 'I:\Amr\Blender\Federate\Site Viewer\SiteViewer\bin\Debug\Cosye.Industrial.dll', Symbols loadec
Cosye.FederateHost.exe' (Managed): Loaded 'I:\Amr\Blender\Federate\Site Viewer\SiteViewer\bin\Debug\SiteViewer.dll', Symbols loaded.
Cosye.FederateHost.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Cosye.Hla.Rti\3.5.1.0__0870fe8ecaba8728\Cosye.Hla.Rti.dll'
Cosye.FederateHost.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualBasic\8.0.0.0__b03f5f7f11d50a3a\Microsoft.VisualE
```

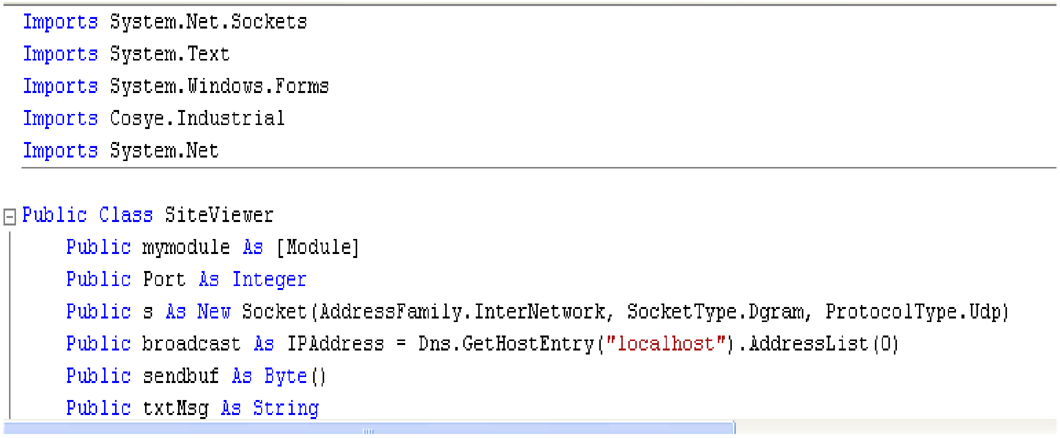
Properties

MyModuleFactory Cosye.Industr

EarlyStart	Neither
⊞ FederateAmbassad	**fedAmb**
FieldLocation	**Subscribe**
GenerateMember	True
ID	Neither
Length	Neither
LiftTime	Neither
LocationID	Neither
Men	Neither
Modifiers	**Friend**
Name	**Subscribe**

Figure 4 Site Viewer List Box to show msgs from Cosye.Industrial plus subscribing in various attributes in MyModuleFactory

As shown in Figure 4 that a copy of the FOM is part of the site.viewer solution (called Cosye.Industrial.xml). Now, once the time regulation and public/subscribe to simulation's objects classes' attributes is complete, it is time to develop the VB.Net code for reflecting the messages of updated attribute values from the Source Simulation.



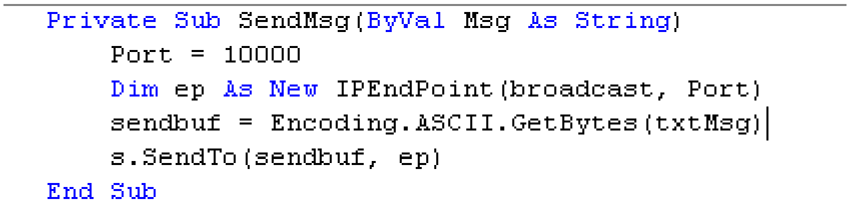
```
Imports System.Net.Sockets
Imports System.Text
Imports System.Windows.Forms
Imports Cosye.Industrial
Imports System.Net

Public Class SiteViewer
    Public mymodule As [Module]
    Public Port As Integer
    Public s As New Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp)
    Public broadcast As IPAddress = Dns.GetHostEntry("localhost").AddressList(0)
    Public sendbuf As Byte()
    Public txtMsg As String
```

Figure 5 The Imports part and Declarations before getting into the coding

As show in figure 5:

- mymodule as module to declare an instance of the modules published by the Cosye.Industrial federation.
- S as socket in Port = Port which reads from a certain IP address (broadcast) , which is set here to Localhost, this IP address can change based on the IP address of the terminal where the Site Viewer federate is located.
- txtMsg is the message that this Site.Viewer will be sending to the BGE.

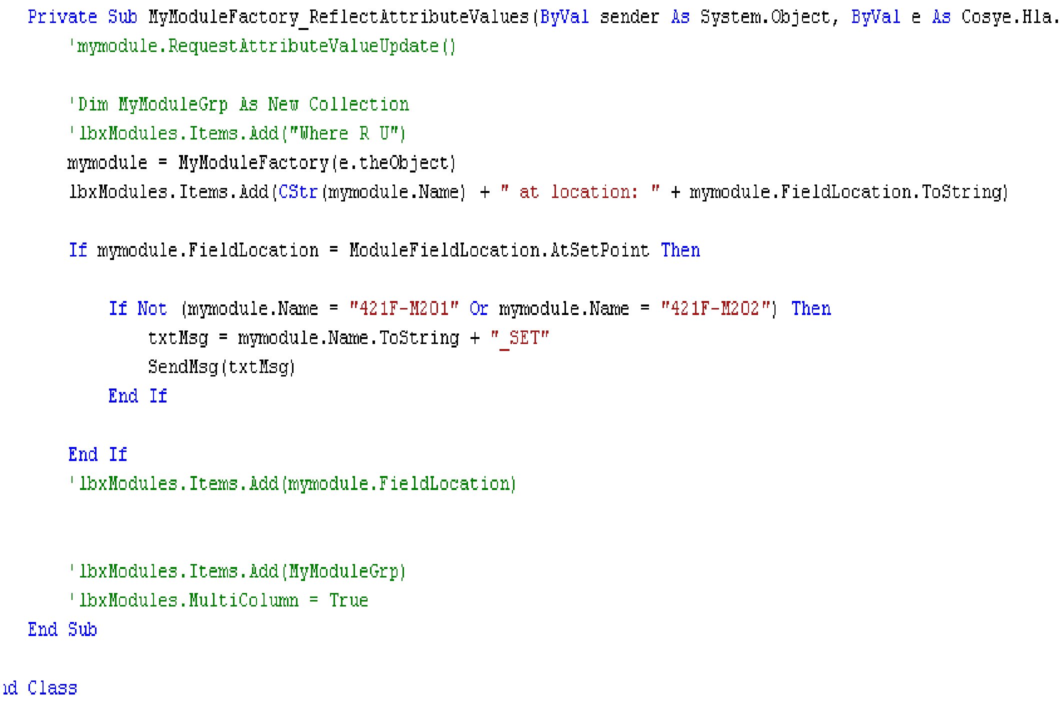
How do we connect the Site Viewer (Client) to the BGE (server)?



```
Private Sub SendMsg(ByVal Msg As String)
    Port = 10000
    Dim ep As New IPEndPoint(broadcast, Port)
    sendbuf = Encoding.ASCII.GetBytes(txtMsg)
    s.SendTo(sendbuf, ep)
End Sub
```

Figure 6 The Send Message function

The SendMsg function simply sends the message to the BGE, the msg in this case will be the mymodle.name plus it's FieldLocation, which is basically the reflection that the Site Viewer receives from the Cosye.Industrial Federation. The SendMsg function uses Port=10000 (Port was declared in Figure 5), the IP address (broadcast = localhost), and the port 10000 inside an address (EndPoint) called ep. Now socket (s) sends the msg to this address (ep) using this sendbuf (buffer). Now, after the entire send message was set, the message has to be defined (to represent module name and Field location) in this case.



```
Private Sub MyModuleFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As Cosye.Hla.
    'mymodule.RequestAttributeValueUpdate()

    'Dim MyModuleGrp As New Collection
    'lbxModules.Items.Add("Where R U")
    mymodule = MyModuleFactory(e.theObject)
    lbxModules.Items.Add(CStr(mymodule.Name) + " at location: " + mymodule.FieldLocation.ToString)

    If mymodule.FieldLocation = ModuleFieldLocation.AtSetPoint Then

        If Not (mymodule.Name = "421F-M201" Or mymodule.Name = "421F-M202") Then
            txtMsg = mymodule.Name.ToString + "_SET"
            SendMsg(txtMsg)
        End If

    End If
    'lbxModules.Items.Add(mymodule.FieldLocation)


    'lbxModules.Items.Add(MyModuleGrp)
    'lbxModules.MultiColumn = True
End Sub

nd Class
```

Figure 7 sending a message (txtMsg) each time a reflection is received from the RTI

Figure 7 shows what happens when there is a reflection from the Cosye.Industrial Federation through the RTI, first setting the instance we have (mymodule) to the object of which its attributes are reflected:

mymodule= mymodulefactory(e.theobject)

Next it is the filtering of the modules such that a message is only sent using the sendmsg function if a module is AtSetPoint, and it is not one of two modules that are not represented in the BGE site model with an object. The txtMsg is then set to Module Name + "SET", this will be truncated at the receiver's end (BGE) through the slicing function to send a message called ("SET") inside the BGE model, hence the case sensitivity of the BGE python scripts. The sendmsg function with its parameters set earlier in Figure 6, can be used to send this message on the specified IP and Port (Localhost & 10000 respectively in this case), the BGE model will take it from there as a receiver.

Now it is time to move to the last component of the HSV framework which is the BGE model with its graphical and python interface, which is where:

1. The sent message will be received

2. Based on this message the modules will be demonstrated at their respective site location and the cranes will move to their respective locations.

3rd: BGE Site Model:

The logic inside the gaming engine is built in general using 3 blocks:

- sensors : That is usually where it receives a message
- controllers: where the python code lays
- actuators: starts certain action

The BGE interface will be broken down into 2 major components:

- BGE site model acting as a server to receive and send)messages to the Site Viewer Federate
- Internal BGE logic through Logic Bricks and Python scripts to insert the action based on messages received from Site Viewer Federate.

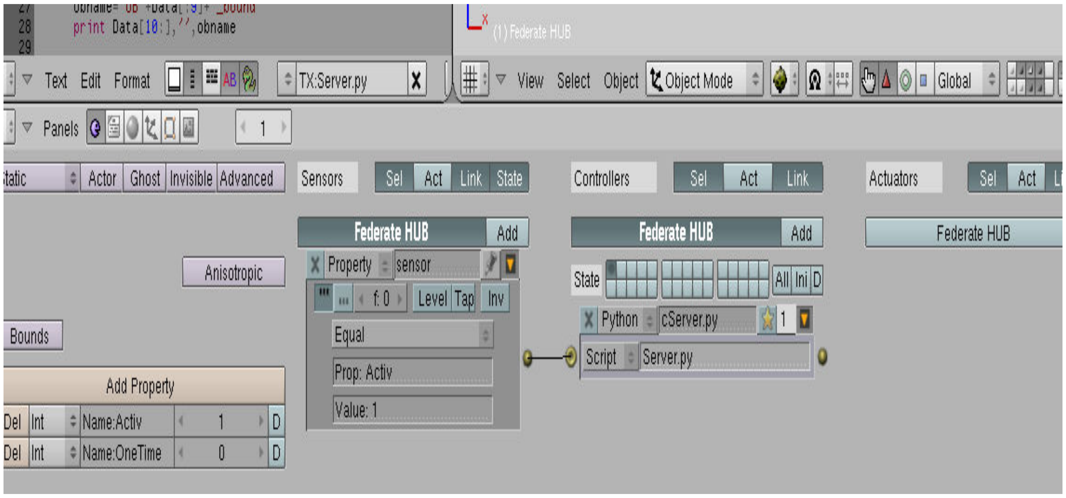
We start in the logical order from receiving messages from the Site Viewer Federate

BGE site model acting as a server to receive and send messages to the Site Viewer Federate

In order to do that a new logic cube was created and called "Federate HUB". The Federate HUB had a property which is called active which was given the type integer (Int) and always set to 1, to act like an always sensor so as to remain active all the time of the simulation run. We want the federate HUB object to remain active all the time because simply you don't know when the message will be sent from the site Viewer so you want always your Federate HUB object to stay on alert all the time. Now, this being active all the time activates the Server.py script on the controller block of the Federate HUB object. See Figure 8 below.

This Server.py is simply the "bottle neck" for all this HSV framework, that is where the BGE Site Model acts as a server and client, it activates the gaming logic by sending the SET msg to send the specified modules defined by Cosye.Industrial to go to their Set Point.

How this happens? Let’s examine the server.py python script.



```
print Data[10:],'',obname
```

Property sensor: sensor, f: 0, Equal, Prop: Activ, Value: 1

Python controller: cServer.py, Script: Server.py

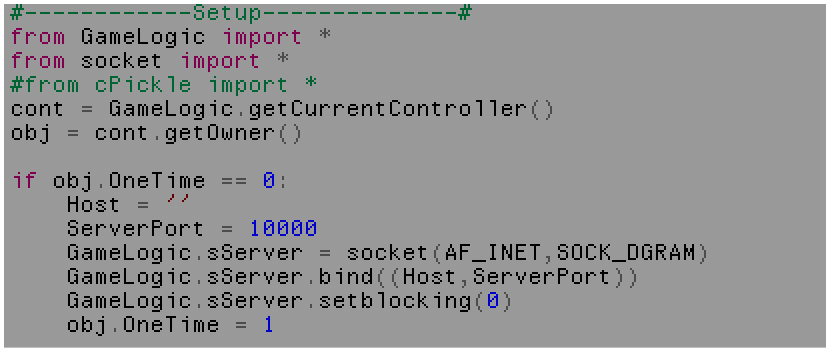
Add Property:

Del	Int	Name:Activ	1
Del	Int	Name:OneTime	0

Figure 8 Federate HUB logic bricks and activating the Server.py script

Now for the server.py, let's analyze the code step by step. The server.py performs basically two functions:

1. Receives the message from the Site Viewer federate(Figure 9):



```
#------------Setup--------------#
from GameLogic import *
from socket import *
#from cPickle import *
cont = GameLogic.getCurrentController()
obj = cont.getOwner()

if obj.OneTime == 0:
    Host = ''
    ServerPort = 10000
    GameLogic.sServer = socket(AF_INET,SOCK_DGRAM)
    GameLogic.sServer.bind((Host,ServerPort))
    GameLogic.sServer.setblocking(0)
    obj.OneTime = 1
```

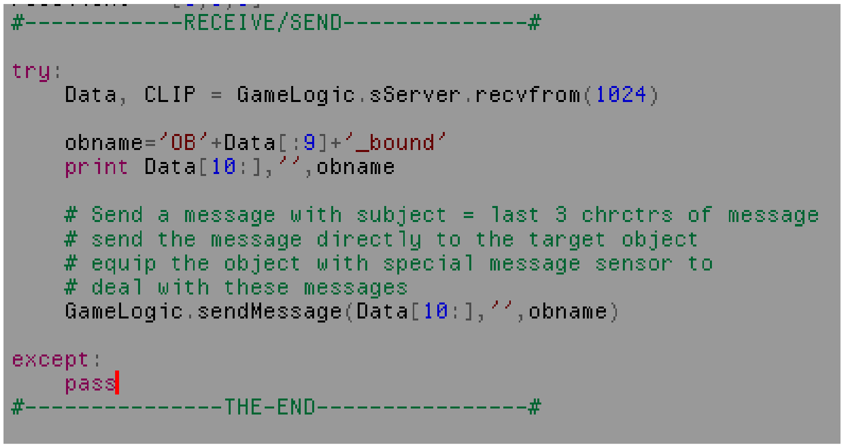
Figure 9 Server.py; receiving message and server connection

Always start with cont = GameLogic.getCurrentController( ) that is where you get the controller of the object you are "standing on". This should always again be followed by getting the Owner of this controller through obj=cont.getOwner ( )

Set the port to the value you specified earlier in the Site Viewer, so that is the port of exchange between the two components. Define the socket as GameLogic.sServer and then bind this socket to the Host and the ServerPort. Host is automatically chosen based on where the message is received that is why it is left blank; Host=' '. Once the connection is set, it is time to define the message we need to send (instead of putting it on a message actuator) and define who should receive this message.

2. Send message to the site viewer federate (Please refer to Chapter 4).

3. Send the message to the predefined object (message = SET , Message Receptor = Bound Object in this case-Figure 10).



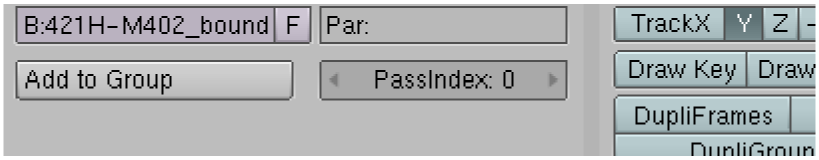
```
#------------RECEIVE/SEND--------------#

try:
    Data, CLIP = GameLogic.sServer.recvfrom(1024)

    obname='OB'+Data[:9]+'_bound'
    print Data[10:],'',obname

    # Send a message with subject = last 3 chrctrs of message
    # send the message directly to the target object
    # equip the object with special message sensor to
    # deal with these messages
    GameLogic.sendMessage(Data[10:],'',obname)

except:
    pass
#---------------THE-END----------------#
```

Figure 10 Server.py; Composing Message and sending it

Now the port is known, so as the IP and the socket is ready. This guarantees that the message will be received from Site Viewer with the Module Name & its Field Location as described earlier. Now this message needs to be send to a specific object inside the BGE Site Model (Module Bound Object in this case). First we define the object name by doing the following:

obname= 'OB'+Data[:9]+'_bound'

This simply sets the object to receive the msg to be equal the Bound Name which starts with OB then the message received from site viewer w/o the (SET) portion, hence the [:9] slicing plus the word bound , add this up and it matched the bound object name.(Figure 11).



B:421H-M402_bound

Par:

PassIndex: 0

Figure 11 Bound Object Name (The OB421H-M402_bound)

For example in Figure 11, the module 421H-M402_SET message will be sliced before the 9th position to result :421H-M402, then add to that the OB as a prefix and the _bound as a suffix and you get the bound object's name where the message SET (in this case) is sent.

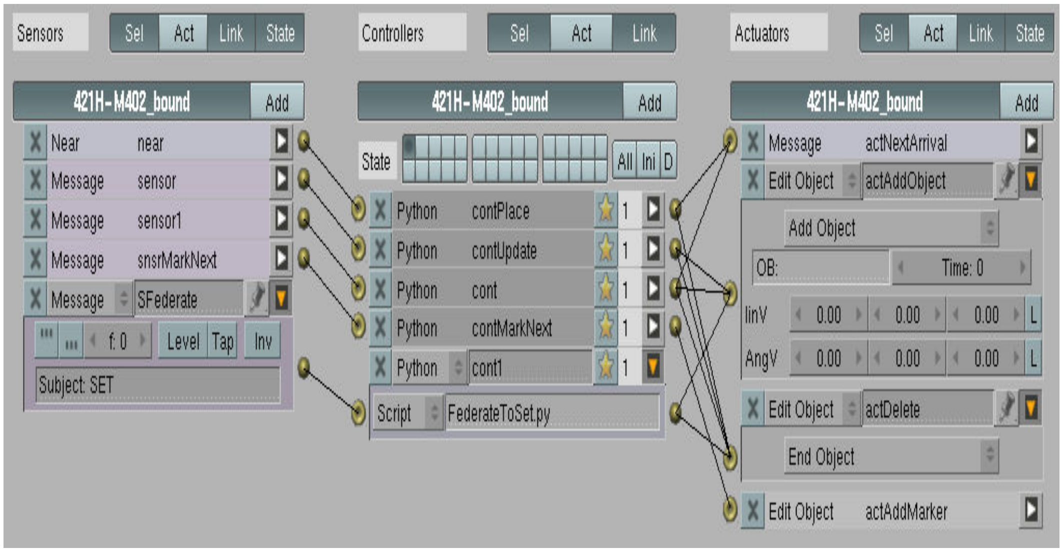
N.B. The semicolons in BGE Python don't count as a space after OB:

After that only sending the message is left, so the message (Called Data in Figure 10.) is sliced after 10 to yield the command message “SET” which is sent to the 'obname' which as explained earlier.

Now the message is sent to each bound object so what next ? It is all now up to BGE site model game scene, what will be triggered inside, this moves us to the second part.

3-2: Internal BGE logic through Logic Bricks and Python scripts to insert the action based on messages received

Now the message is send to the Bound Object for each module respectively that is where we stopped, nd we take it from the same point here. Figure 12 explains that.



421H-M402_bound
Near near
Message sensor
Message sensor1
Message snsrMarkNext
Message SFederate
f: 0 Level Tap Inv
Subject: SET

421H-M402_bound
State
Python contPlace 1
Python contUpdate 1
Python cont 1
Python contMarkNext 1
Python cont1 1
Script FederateToSet.py

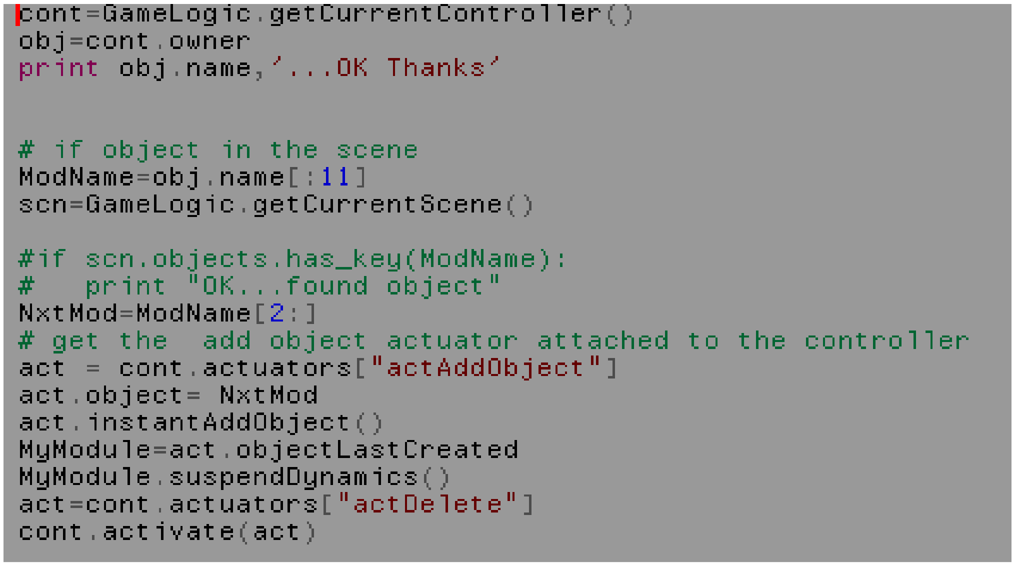
421H-M402_bound
Message actNextArrival
Edit Object actAddObject
Add Object
OB: Time: 0
linV 0.00 0.00 0.00 L
AngV 0.00 0.00 0.00 L
Edit Object actDelete
End Object
Edit Object actAddMarker

Figure 12 Logic inside the BGE after receiving the respective msg from Viewer Federate through the Server.py

The “SET” message is received from Federate HUB object and fires the SFederate sensor as shown in Figure 12. This in turn starts the FederateToSet.py which (through the FederateToSet.py script) fires 2 actuators:

1. actAddObject : Adds the specified module to its bound
2. actDelete : Deletes the Module's respective bound object to minimize the cluttering and suspends the physics engine dynamics.

So the FederateToSet.py script is the corner stone here; but how does the code in this script work.



```
cont=GameLogic.getCurrentController()
obj=cont.owner
print obj.name,'...OK Thanks'


# if object in the scene
ModName=obj.name[:11]
scn=GameLogic.getCurrentScene()

#if scn.objects.has_key(ModName):
#   print "OK...found object"
NxtMod=ModName[2:]
# get the  add object actuator attached to the controller
act = cont.actuators["actAddObject"]
act.object= NxtMod
act.instantAddObject()
MyModule=act.objectLastCreated
MyModule.suspendDynamics()
act=cont.actuators["actDelete"]
cont.activate(act)
```

Fig 13: FederateToSet.py script

This script basically does 2 things:

1. Finds the module from the first hidden layer with the OB prefix to module name [slice the bound object :11], this gets the module name with "ob" and recalls it from hidden layer. Next, the code adds this object to the current scene by activating the 'actAddObject' and defining the module name w/o the OB to be the object added (NxtMod with ModName sliced after 2) as the object to be added.
2. This is followed by removing this Bound Object to reduce cluttering.

N.B. This FederateToSet.py is developed with the help of the ToSetPoint.py script.

N.B The resources (e.g. cranes objects...etc.) objects' simulation statuses to visualization behavior developments were done through the same steps mentioned above for the modules.

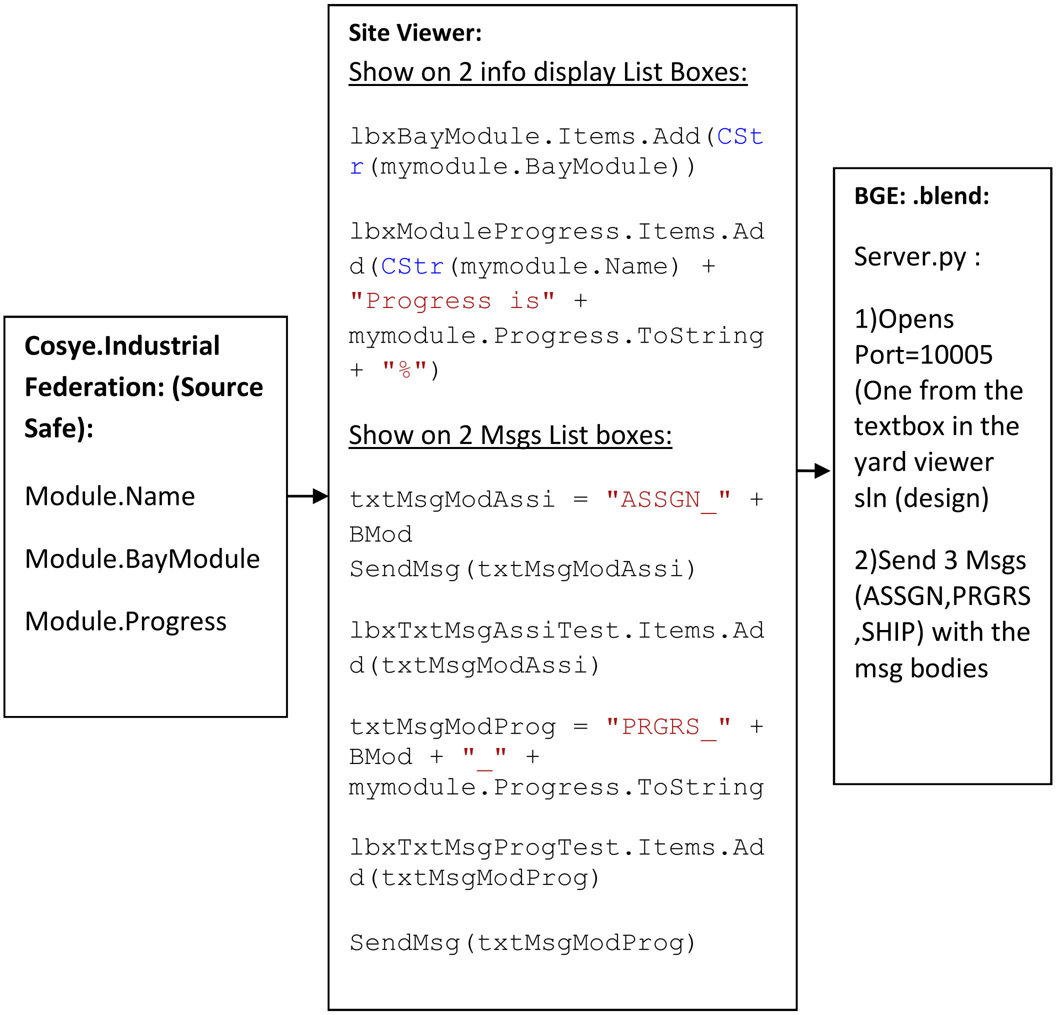
Appendix III: Yard Visualization Manual

The development of the yard visualization federate and the industrial federation was done using Cosye framework which is a C# framework that allows developers to develop interoperable simulations using HLA standards. Cosy was developed by programming team at the University of Alberta's Hole School of Construction Engineering and it can be downloaded from http://irc.construction.ualberta.ca/cosye. The site and yard interactive visualization screens were developed through Blender©, which is an open source 3D engine with a gaming engine known as Blender Game Engine (BGE). Blender can be downloaded from this website: www.blender.org .

This manual is to further explain the technical details of the yard visualization federate and site visualization Blender Game engine (BGE) important developments. The manual does not include all the developments' details; however it includes enough to put the reader on the right track to follow through this development for future simulation visualization developments using the HSV framework's components.

Visualization Pipeline Mechanism:

The way the logic is designed is such that a certain object inside the scene of the game (yard blend file) acts as a the Server. The object is: "Server"-an invisible object-and it is the server which has the Server.py on its controller. While the Client in this case is the Yard Viewer VB.NET solution (Yard Viewer Federate). This client (Yard Viewer Federate) sends reflections of Module.BayModule , Module.Name & Module.Progress from the Construction Industrial federation. The way it goes is as follows:



Cosye.Industrial Federation: (Source Safe):

Module.Name

Module.BayModule

Module.Progress

Site Viewer:

Show on 2 info display List Boxes:

```
lbxBayModule.Items.Add(CStr(mymodule.BayModule))

lbxModuleProgress.Items.Add(CStr(mymodule.Name) + "Progress is" + mymodule.Progress.ToString + "%")
```

Show on 2 Msgs List boxes:

```
txtMsgModAssi = "ASSGN_" + BMod
SendMsg(txtMsgModAssi)

lbxTxtMsgAssiTest.Items.Add(txtMsgModAssi)

txtMsgModProg = "PRGRS_" + BMod + "_" + mymodule.Progress.ToString

lbxTxtMsgProgTest.Items.Add(txtMsgModProg)

SendMsg(txtMsgModProg)
```

BGE: .blend:

Server.py :

1)Opens Port=10005 (One from the textbox in the yard viewer sln (design)

2)Send 3 Msgs (ASSGN,PRGRS,SHIP) with the msg bodies

Figure 1 The logical flow of one way of the HSV framework pipeline

1st: The Cosye.Industrial Federation:

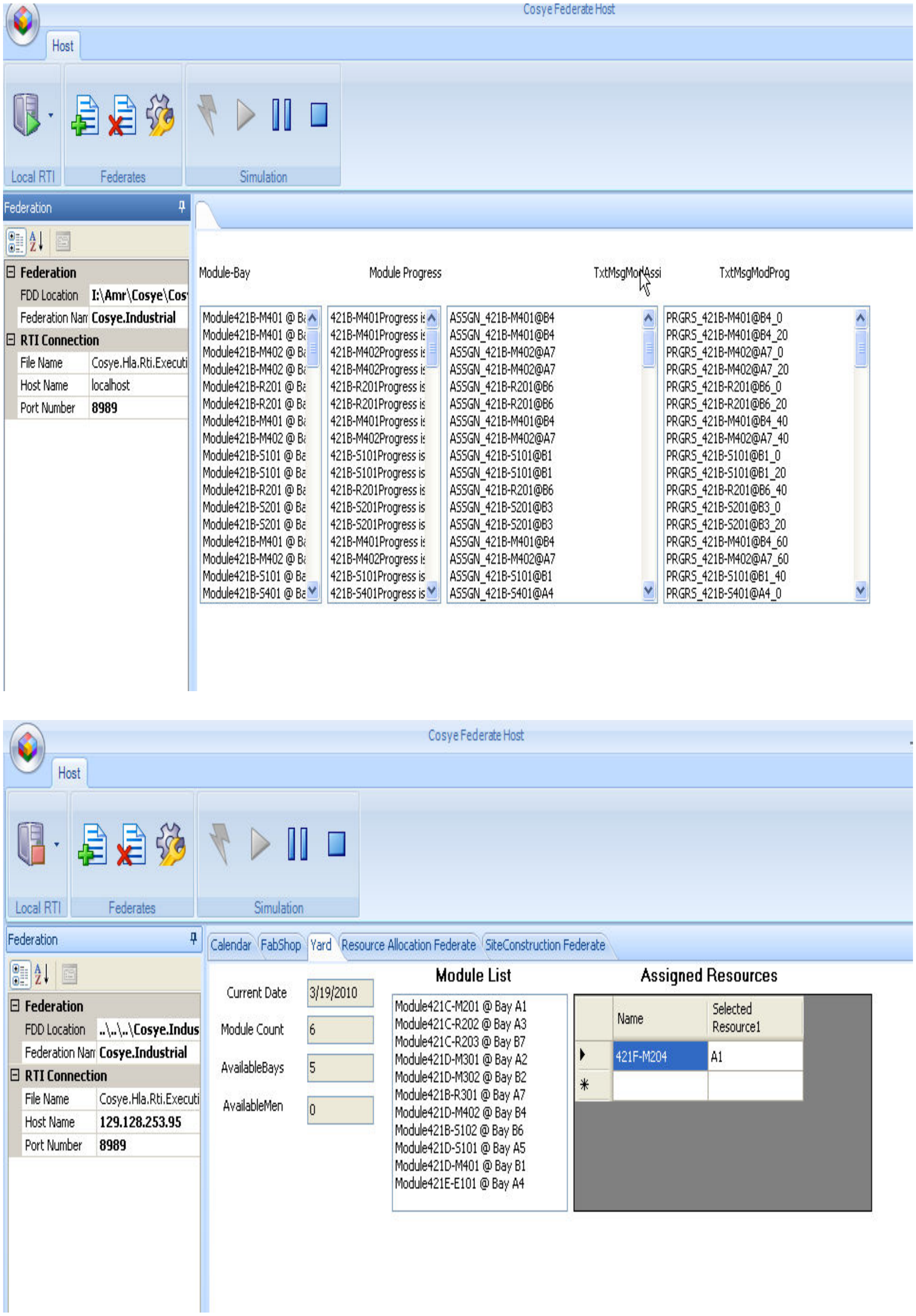
As it was explained in Site Viewer Federate manual (Appendix II), except that it publishes other attributes that are read by the Yard Viewer Federate:

Module.Name: Has a format of : NNNC-CNNN , For Example: 421B-M401

Module.BayModule: Has a format of : ModuleNNNC-CNNN @ Bay CN, for example : Module421C-M201 @ Bay A1

Module.Progress: NN , for example 70

where N = Integer
C= Alphabetic Character



⊟ **Federation**	
FDD Location	**I:\Amr\Cosye\Cos**
Federation Nan	**Cosye.Industrial**
⊟ **RTI Connection**	
File Name	Cosye.Hla.Rti.Executi
Host Name	localhost
Port Number	**8989**

Module-Bay	Module Progress	TxtMsgModAssi	TxtMsgModProg
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_0
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_20
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_0
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_20
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_0
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_20
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_40
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_40
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_0
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_20
Module421B-R201 @ Ba	421B-R201Progress is	ASSGN_421B-R201@B6	PRGRS_421B-R201@B6_40
Module421B-S201 @ Ba	421B-S201Progress is	ASSGN_421B-S201@B3	PRGRS_421B-S201@B3_0
Module421B-S201 @ Ba	421B-S201Progress is	ASSGN_421B-S201@B3	PRGRS_421B-S201@B3_20
Module421B-M401 @ Ba	421B-M401Progress is	ASSGN_421B-M401@B4	PRGRS_421B-M401@B4_60
Module421B-M402 @ Ba	421B-M402Progress is	ASSGN_421B-M402@A7	PRGRS_421B-M402@A7_60
Module421B-S101 @ Ba	421B-S101Progress is	ASSGN_421B-S101@B1	PRGRS_421B-S101@B1_40
Module421B-S401 @ Ba	421B-S401Progress is	ASSGN_421B-S401@A4	PRGRS_421B-S401@A4_0

⊟ **Federation**	
FDD Location	**..\..\..\Cosye.Indus**
Federation Nan	**Cosye.Industrial**
⊟ **RTI Connection**	
File Name	Cosye.Hla.Rti.Executi
Host Name	**129.128.253.95**
Port Number	**8989**

Current Date 3/19/2010

Module Count 6

AvailableBays 5

AvailableMen 0

Module List

Module421C-M201 @ Bay A1
Module421C-R202 @ Bay A3
Module421C-R203 @ Bay B7
Module421D-M301 @ Bay A2
Module421D-M302 @ Bay B2
Module421B-R301 @ Bay A7
Module421D-M402 @ Bay B4
Module421B-S102 @ Bay B6
Module421D-S101 @ Bay A5
Module421D-M401 @ Bay B1
Module421E-E101 @ Bay A4

Assigned Resources

	Name	Selected Resource1
▶	421F-M204	A1
*		

Figure 2 the Site Viewer Federate (Client) and the Cosye.Industrial Federation Federate Hosts

2nd: The Yard Viewer Federate-Client:

The Site Viewer Federate is now to receive reflections from the Industrial Federation (which may run on same or different terminal with a different IP address). The receiving of reflections here will be through the COSYE.RTI not through messages. The YardViewer is only sending message to the BGE's Server object, which acts as a server to the Yard Viewer client. This is very important to understand, that federates contacts are through the RTI on the same terminal as the Yard Viewer client.

Please check the *Site Viewer Federate Manual (Appendix II)* for:

1. Creating the host federate and connecting it to the RTI
2. Imports portion of the YardViewer code.

Now for the messages part; there are 2 messages that are sent with the following formats:

txtMsgModAssi:

ASSGN_ NNNC-CNNN@CN

Truncated BayModule attribute of original format: ModuleNNNC-CNNN @ Bay CN

Truncation to reach this form is through the following code:

```
BMod = mymodule.BayModule.ToString
BMod = BMod.Remove(0, 7)
```

This line removes the "Module" portion from ModuleNNNC-CNNN @ Bay CN, so the result is NNNC-CNNN @ Bay CN.

```
BMod = BMod.Remove(9, 1)
```

This line removes the "space" portion from NNNC-CNNN @ Bay CN, so the result is NNNC-CNNN@ Bay CN. It reads from index # 9 which is the N character (last N in CNNN).

```
BMod = BMod.Remove(10, 5)
```

This line removes the "space after @+Bay+space after Bay" from NNNC-CNNN@ Bay CN, so the result is NNNC-CNNN@CN. It reads from index # 10 which is the @ character and counts 5 after it.

```
txtMsgModAssi = "ASSGN_" + BMod
```

Now resulting : "ASSGN_NNNC-CNNN@CN" to the port # 10005 or the one defined in the code (`Port = PortSpinBox.Value`)

txtMsgModProg:

```
txtMsgModProg = "PRGRS_" + BMod + "_" + mymodule.Progress.ToString
```

This line results in:PRGRS_ added to the resulting BMod (NNNC-CNNN@CN)+_+NN

Now, the 2 messages are sent to the BGE which has both the module and the bay objects. Let's see the code that would recall the Module from a hidden layer and make it move to Bay then grow there based on its percentage growth coming from the txtMsgModProg message

3rd: The Yard Viewer Model in the BGE:

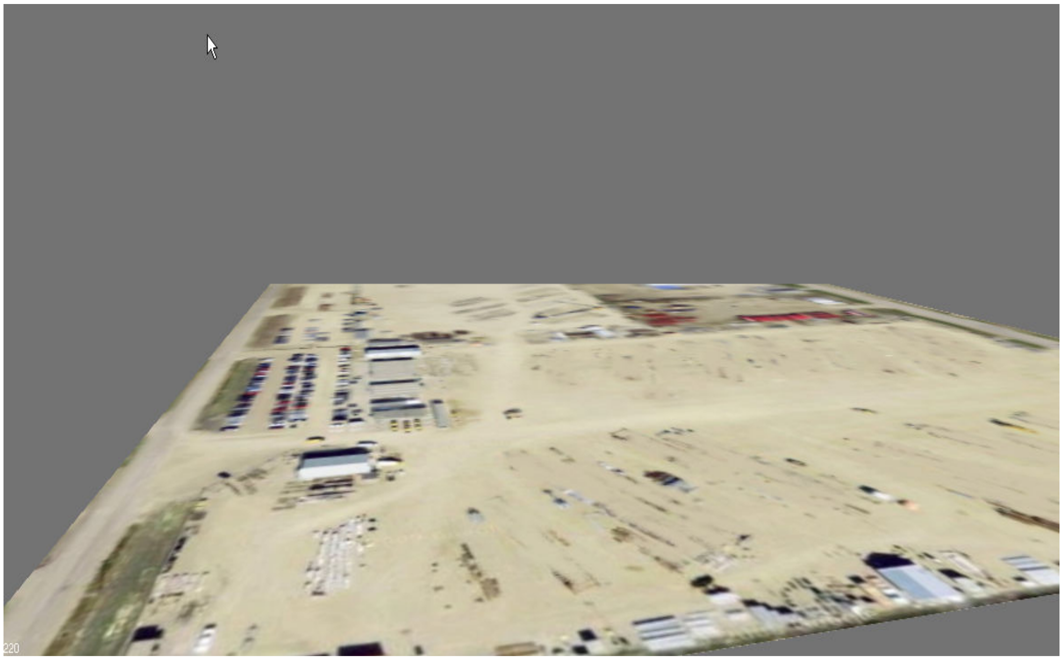
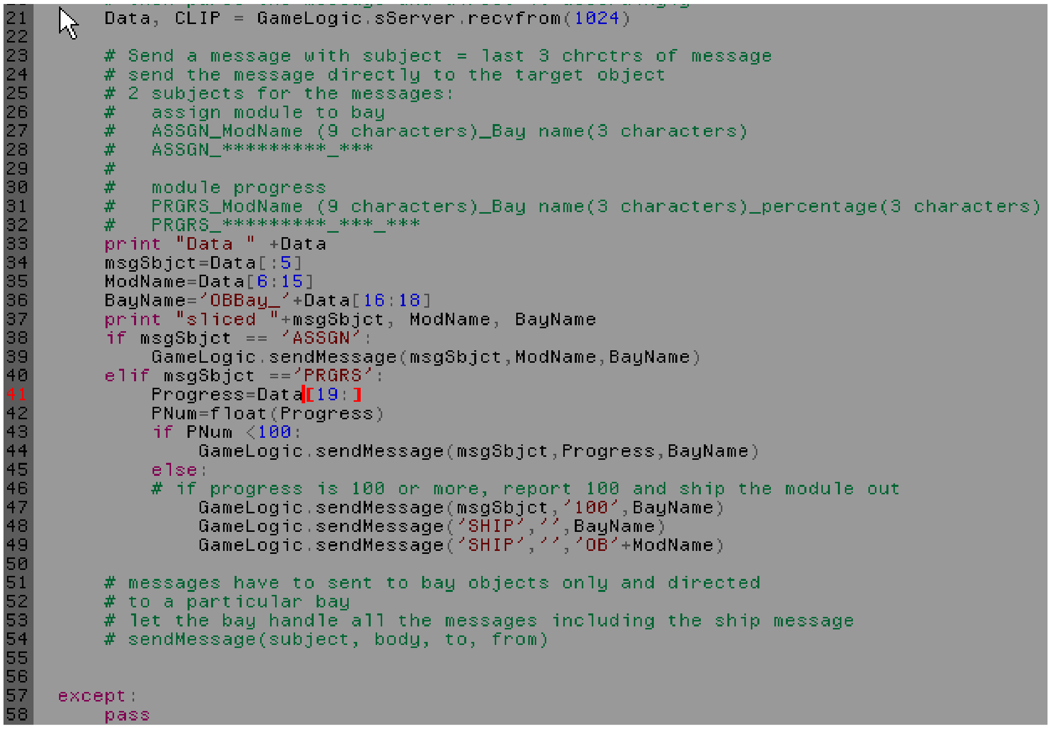


Figure 3 the Yard Federate with a real picture as background before the modules are assigned to the bay objects through the BGE.

Now the 2 messages are sent from the Yard Viewer ffederate will be received through the server.py :



```
    Data, CLIP = GameLogic.sServer.recvfrom(1024)

    # Send a message with subject = last 3 chrctrs of message
    # send the message directly to the target object
    # 2 subjects for the messages:
    #   assign module to bay
    #   ASSGN_ModName (9 characters)_Bay name(3 characters)
    #   ASSGN_*********_***
    #
    #   module progress
    #   PRGRS_ModName (9 characters)_Bay name(3 characters)_percentage(3 characters)
    #   PRGRS_*********_***_***
    print "Data " +Data
    msgSbjct=Data[:5]
    ModName=Data[6:15]
    BayName='OBBay_'+Data[16:18]
    print "sliced "+msgSbjct, ModName, BayName
    if msgSbjct == 'ASSGN':
        GameLogic.sendMessage(msgSbjct,ModName,BayName)
    elif msgSbjct =='PRGRS':
        Progress=Data[19:]
        PNum=float(Progress)
        if PNum <100:
            GameLogic.sendMessage(msgSbjct,Progress,BayName)
        else:
        # if progress is 100 or more, report 100 and ship the module out
            GameLogic.sendMessage(msgSbjct,'100',BayName)
            GameLogic.sendMessage('SHIP','',BayName)
            GameLogic.sendMessage('SHIP','','OB'+ModName)

    # messages have to sent to bay objects only and directed
    # to a particular bay
    # let the bay handle all the messages including the ship message
    # sendMessage(subject, body, to, from)


except:
    pass
```

Figure 4 Server.py sending messages and assigning names

The message subject is first extracted by slicing at 5: msgSbjct=Data[:5] which results in two types of msgSbjct: ASSGN & PRGRS based on slicing ASSGN_NNNC-CNNN@CN and PRGRS_ NNNC-CNNN_NN.

Now the server.py code on will issue two things from the any message it receives (ASSGN or PRGRS):

ModName=NNNC-CNNN [6:15]

BayName= CN [16:18], then add OBBay_ to it

code: ModName=Data[6:15]; BayName='OBBay_'+Data[16:18]

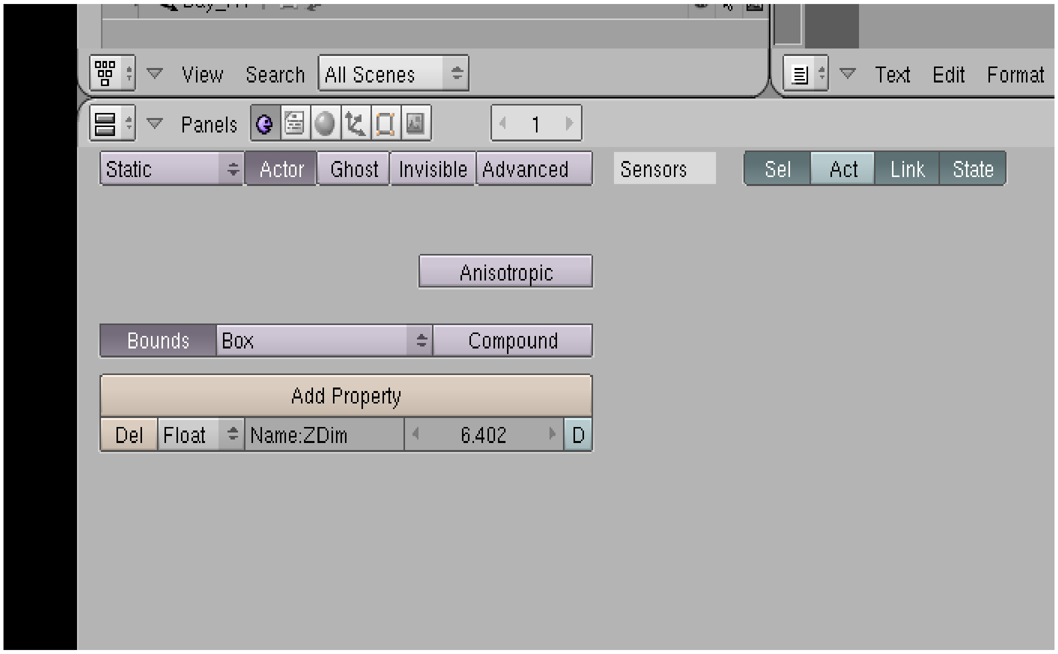
Now after extracting the 3 elements we need ; msgsbjct, ModName & BayName we need to filter the messages based on their type to get the progress % from the PRGRS message then check it should be shipped out in case the progress =100 (SHIP) or it should stay on its respective bay and animate its growth if the progress is less than 100% (PRGRS).

Properties set in Model

Before getting into the 3 types of messages produced, the properties which will be used to animate the progress of a certain module in its respective bay will be discussed.

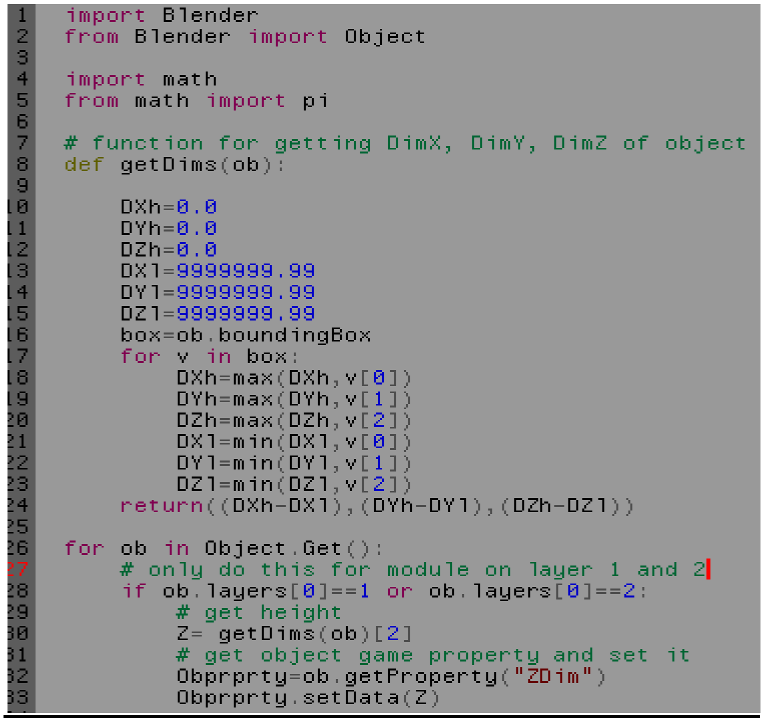
1st Properties set for Modules:

Figure 5 below shows the property (ZDim) assigned to each and every module in the hidden layer. The value of the property (6.402 in this example) is assigned through the python script; GetHeight.py (Figure 6).



Float Name:ZDim 6.402 D

Figure 5 property of module height ZDim



```
import Blender
from Blender import Object

import math
from math import pi

# function for getting DimX, DimY, DimZ of object
def getDims(ob):

    DXh=0.0
    DYh=0.0
    DZh=0.0
    DXl=9999999.99
    DYl=9999999.99
    DZl=9999999.99
    box=ob.boundingBox
    for v in box:
        DXh=max(DXh,v[0])
        DYh=max(DYh,v[1])
        DZh=max(DZh,v[2])
        DXl=min(DXl,v[0])
        DYl=min(DYl,v[1])
        DZl=min(DZl,v[2])
    return((DXh-DXl),(DYh-DYl),(DZh-DZl))

for ob in Object.Get():
    # only do this for module on layer 1 and 2
    if ob.layers[0]==1 or ob.layers[0]==2:
        # get height
        Z= getDims(ob)[2]
        # get object game property and set it
        Obprprty=ob.getProperty("ZDim")
        Obprprty.setData(Z)
```

Figure 6 GetHeight.py to calculate the ZDim property value for each module

getDims function is used to calculate the height of each object, then set its data =z.

Now this property for modules in layers 1 & 2 will be used when a module os in PRGRS or SHIP, now let's check the properties in the small bay objects.

2nd Properties set for Bay Objects:

Figure 7 shows the properties that the bay objects carry and the way that 2 of these properties are assigned values through Property Actuators.

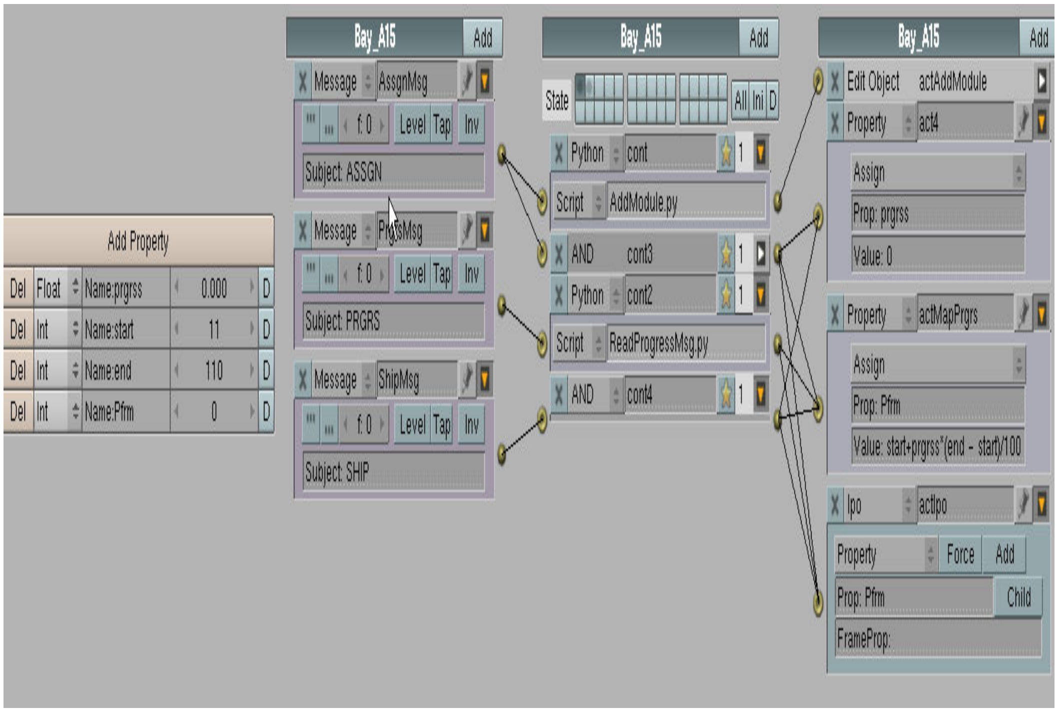


Figure 7 The typical properties that the Bay Objects are assigned and the Property Actuator setting the values for the ***prgrss & Pfrm*** properties

Each Bay Object is assigned 4 properties as shown in the example in figure 7. The Start & End properties had a default value of 11 & 100 respectively, however they do change to match each module's height, through this code in AddModule.py:

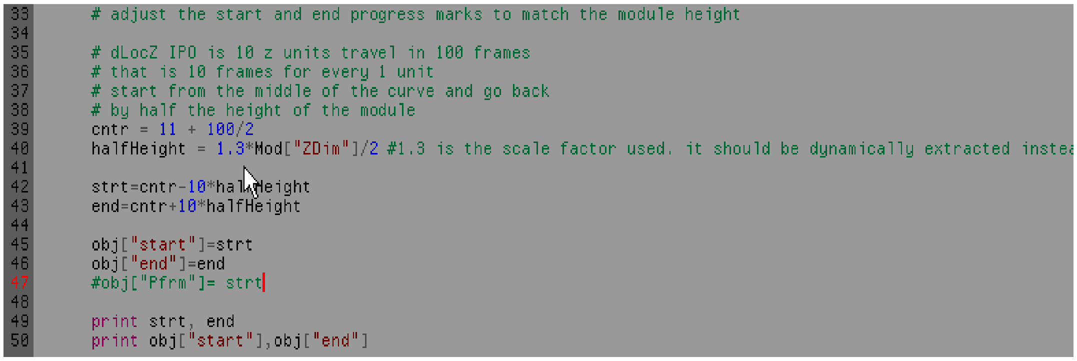

```
    # adjust the start and end progress marks to match the module height

    # dLocZ IPO is 10 z units travel in 100 frames
    # that is 10 frames for every 1 unit
    # start from the middle of the curve and go back
    # by half the height of the module
    cntr = 11 + 100/2
    halfHeight = 1.3*Mod["ZDim"]/2 #1.3 is the scale factor used. it should be dynamically extracted instea

    strt=cntr-10*halfHeight
    end=cntr+10*halfHeight

    obj["start"]=strt
    obj["end"]=end
    #obj["Pfrm"]= strt

    print strt, end
    print obj["start"],obj["end"]
```

Figure 8 AddModule.py setting the ***start & end*** properties for each module respectively

The ZDim property calculated for each module using the script shown earlier in Figure 6 is used here in the equation to calculate the halfHeight. Once this is calculated a *start & end* properties for the animation is set as conceptually shown in figure 9.

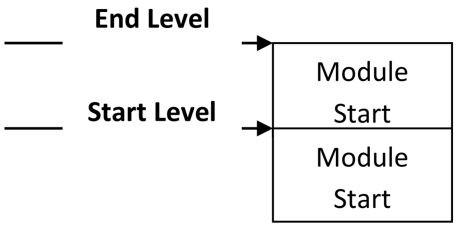


Figure 9 Conceptual Idea behind start level & Finish Level

For the other 2 properties ***(Prgrss & Pfrm***), they both have default values of 0. However, the ***Prgrss*** value is adjusted through the Property Actuator called "act4" which fires when a module is placed in its bay for the first time set the ***prgrss value= 0***. In this case the ***Pfrm value is calculated as follows***:

*= Start (calculated using AddModule.py as in Figure 7) + [prgrss (=0, since the Property Actuator act4 is fired when module is added for first time) *(end(calculated using AddModule.py as in fig. 7)-Start)/100]*

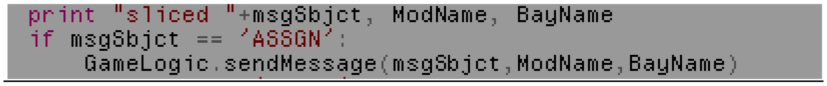
= Start (since prgrss is equal to 0)

Now, let's see how these properties values are played around with to make a module assigned and disappear when it is 100% complete.

<u>Messages and final logic in the BGE:</u>

There are 3 cases with 3 messages "ASSGN","PRGRS" & "SHIP"

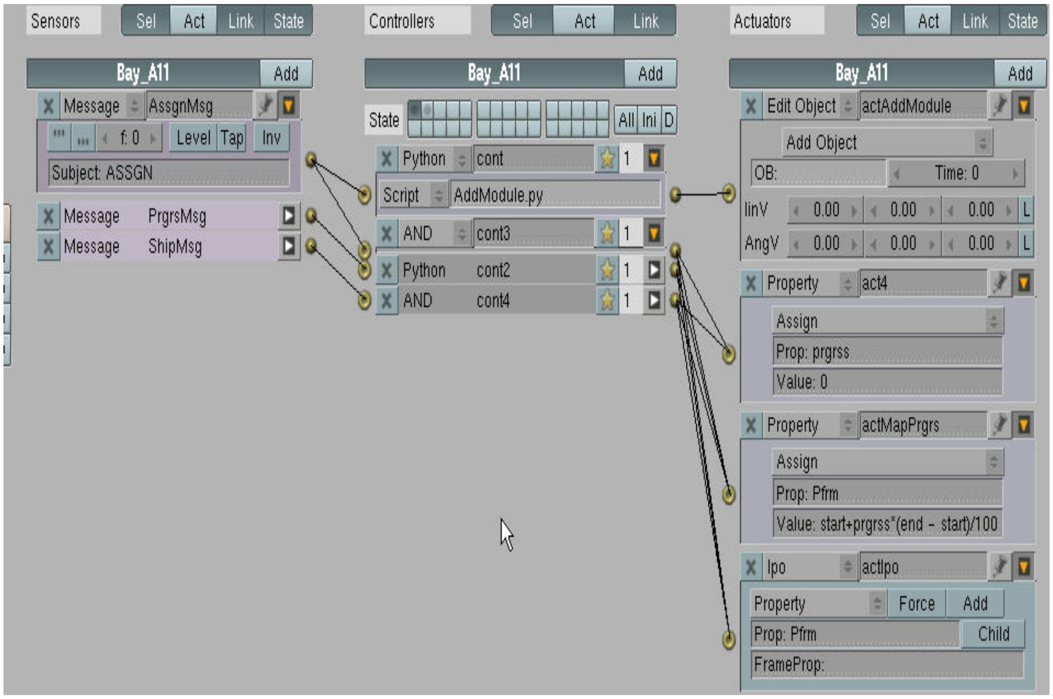
<u>Case 1 the message coming is of the ASSGN & SHIP type:</u>



```
print "sliced "+msgSbjct, ModName, BayName
if msgSbjct == 'ASSGN':
    GameLogic.sendMessage(msgSbjct,ModName,BayName)
```

Figure 10 ASSGN msg from server.py

Now the ASSGN msg is sent to continue with the add module to bay object, so the pipeline is followed as shown in figure 11:



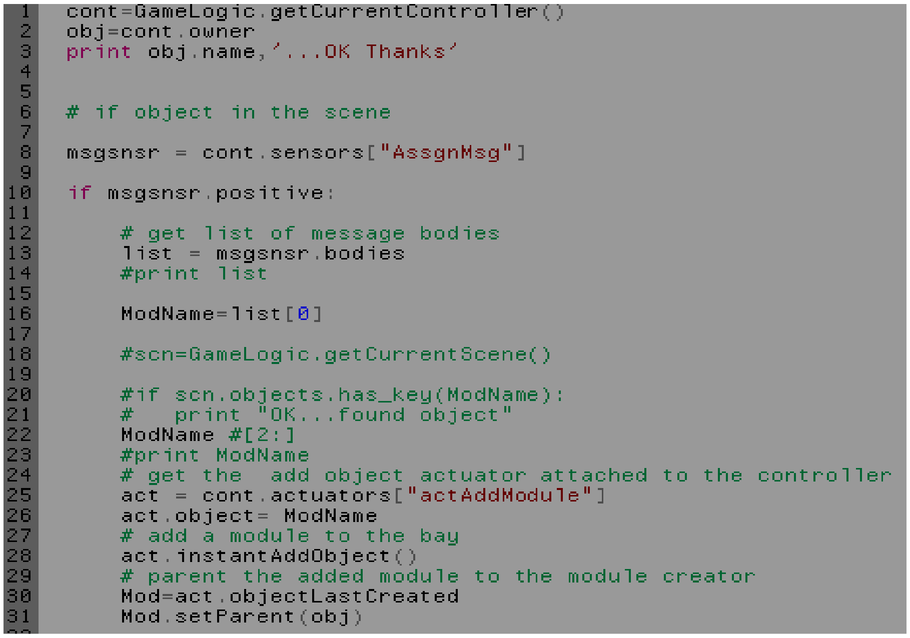
Message AssgnMsg
f: 0
Subject: ASSGN
Message PrgrsMsg
Message ShipMsg

Python cont 1
Script AddModule.py
AND cont3 1
Python cont2 1
AND cont4 1

Edit Object actAddModule
OB: Time: 0
linV 0.00 0.00 0.00
AngV 0.00 0.00 0.00
Property act4
Prop: prgrss
Value: 0
Property actMapPrgrs
Prop: Pfrm
Value: start+prgrss*(end - start)/100
Ipo actIpo
Property
Prop: Pfrm
FrameProp:

Figure 11 The ASSGN msg pipeline

The ASSGN msg is received through the AssgnMsg sensor which then goes to a python controller called cont which has the script AddModule.py (Figure 12), and another AND controller (cont3) that controllers the properties mentioned earlier.



```
cont=GameLogic.getCurrentController()
obj=cont.owner
print obj.name,'...OK Thanks'


# if object in the scene

msgsnsr = cont.sensors["AssgnMsg"]

if msgsnsr.positive:

    # get list of message bodies
    list = msgsnsr.bodies
    #print list

    ModName=list[0]

    #scn=GameLogic.getCurrentScene()

    #if scn.objects.has_key(ModName):
    #   print "OK...found object"
    ModName #[2:]
    #print ModName
    # get the  add object actuator attached to the controller
    act = cont.actuators["actAddModule"]
    act.object= ModName
    # add a module to the bay
    act.instantAddObject()
    # parent the added module to the module creator
    Mod=act.objectLastCreated
    Mod.setParent(obj)
```

Figure 12AddModule.py part for adding module

This code (script) is very similar to the ToSet.py script in the which is part of the Site Viewer and explained in the Site Viewer manual, so please refer to site viewer manual to understand this part (the only difference here is that this AddModule.py script had a message body containing the Module Name in it). Now, the second part of the AddModule.py script will assign values to the start & end properties as shown in figure 7 and explained earlier. These 2 properties will be needed to calculate the Pfrm property which calculates the frame number on which the IPO (animation) will stop based on the prgrss of each module. The equation to decide the frame is as follows in the case that the module is just added:

Pfrm Value= start(from AddModule.py)+ [prgrss(from Property Actuator = 0)(end-start)/100] = Start*

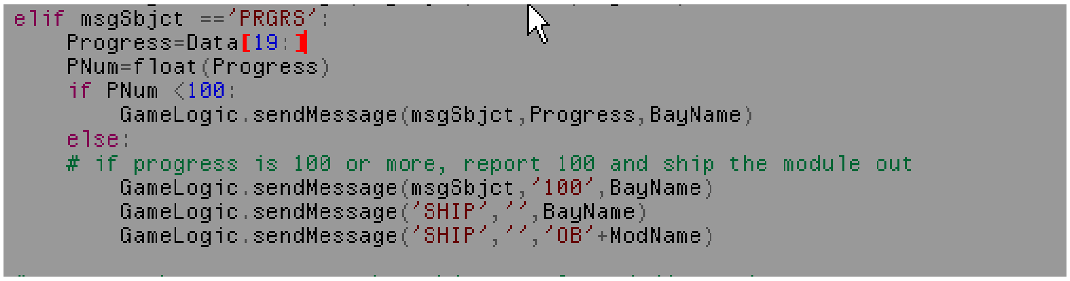
So that is why the ASSGN sensor (AssgnMsg) connects to the AND controller which fires additional three actuators when the module is added to a bay. These 3 actuators are used to calculate the above Pfrm value and start the IPO:

1) Propperty Actuator (act 4) which sets the prgrss to 0 so the value of the Pfrm is = start

2)Property Actuator (actMapPrgrs) which calculates the Pfrm Value as indicated above

3)IPO actuator (actIpo) which starts the IPO and ends at frame# = Pfrm.value = start, based on the above calculation, so the IPO starts and the module is not showing from the picture terrain since its Pfrm (frame number where it stops) is = 0.

when the module is just Assigned or shipped the animation is the same and follows the same concpet, in other words when the prgrss is 0 or 100 then the SHIP message is fired as shown in Figure13 below.

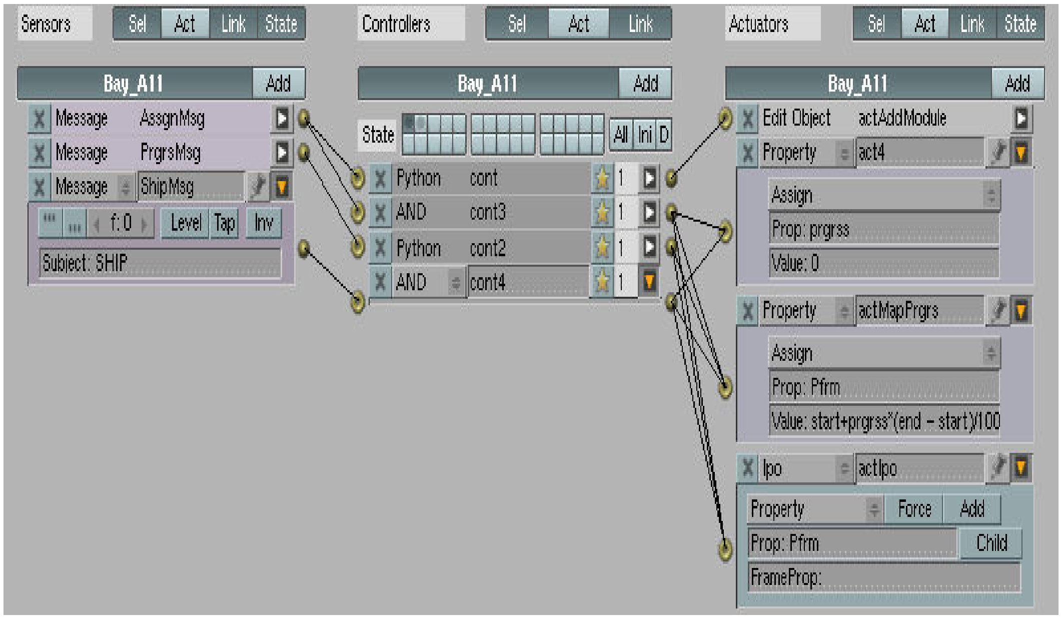


```
elif msgSbjct =='PRGRS':
    Progress=Data[19:]
    PNum=float(Progress)
    if PNum <100:
        GameLogic.sendMessage(msgSbjct,Progress,BayName)
    else:
    # if progress is 100 or more, report 100 and ship the module out
        GameLogic.sendMessage(msgSbjct,'100',BayName)
        GameLogic.sendMessage('SHIP','',BayName)
        GameLogic.sendMessage('SHIP','','OB'+ModName)
```

Figure 13 the SHIP & PRGRS messages sent

So when it is set for shipment i.e. progress =100, then there are 3 messages sent, 2 SHIP msgs and 1 regular PRGRS message. The SHIP messages are connected to the same 3 Property and IPO actuators as it was the case with the ASSGN msg pipeline. This means that it also follows the same logic by setting the prgrs property to 0 and then calculates the pfrm using the pfrm equation above. This sets the pfrm value = start again as it was the case when the module was first added to the bay and had prgrss=0. This simply makes the module disappear as if it is shipped to site. (So it is exactly the same as ASSGN except for the actAddModule actuator that the ASSGN msg pipeline has extra).

Now what does the SHIP msg sent to the module do?



Bay_A11

Message AssgnMsg
Message PrgrsMsg
Message ShipMsg
f: 0
Subject: SHIP

Bay_A11

State
Python cont 1
AND cont3 1
Python cont2 1
AND cont4 1

Bay_A11

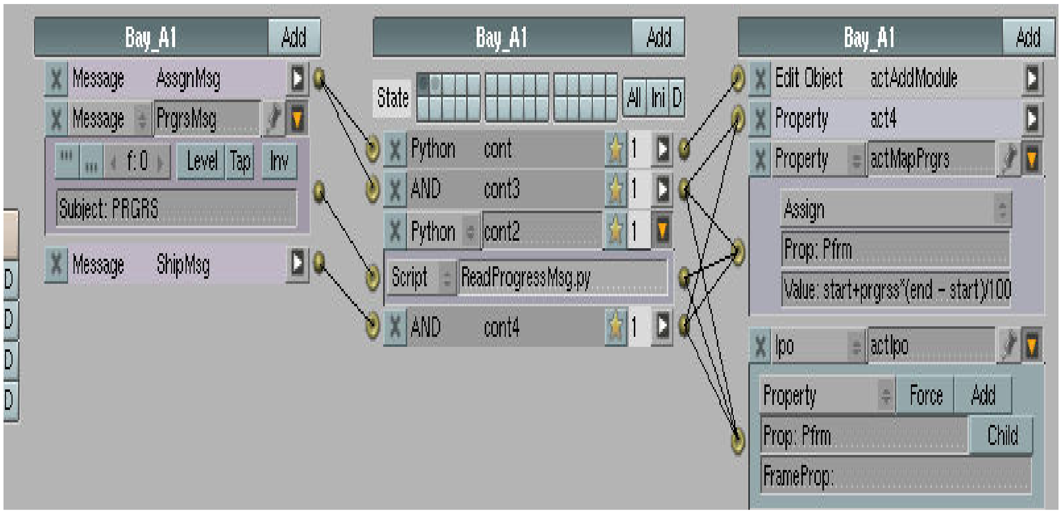
Edit Object actAddModule
Property act4
Assign
Prop: prgrss
Value: 0
Property actMapPrgrs
Assign
Prop: Pfrm
Value: start+prgrss*(end – start)/100
Ipo actIpo
Property
Prop: Pfrm
FrameProp:

Figure 14 SHIP msg logic pipe

On the other hand there is a PRGRS message also sent with prgrss =100 as explained, so what does this msg do? let's see.

Case 2 the message coming is of the PRGRS type:

This msg as shown in figure 12 triggers the PRGRS pipeline shown below in Figure 15.



Bay_A1

Message AssgnMsg
Message PrgrsMsg
f: 0
Subject: PRGRS
Message ShipMsg

Bay_A1

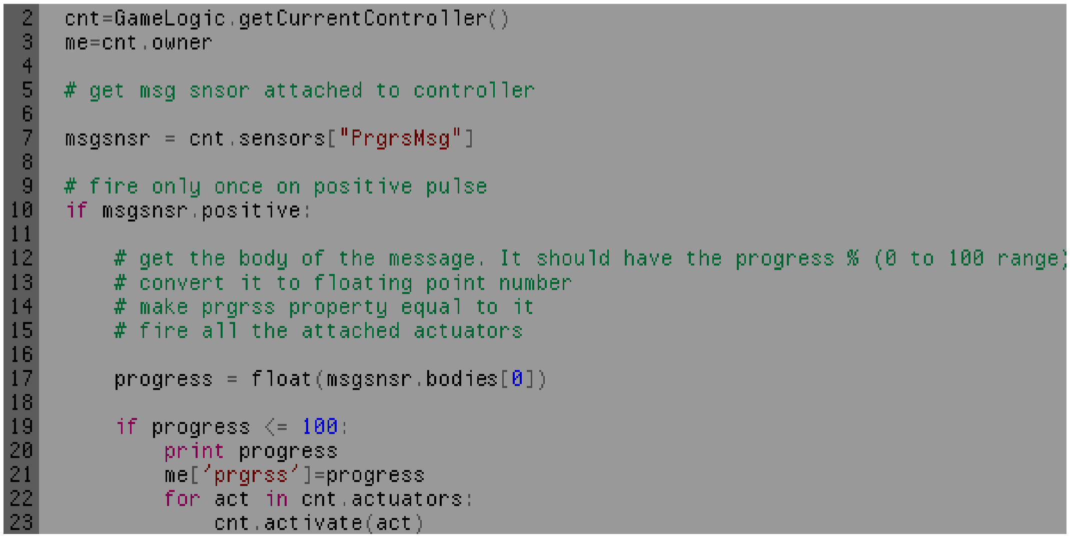
State
Python cont 1
AND cont3 1
Python cont2 1
Script ReadProgressMsg.py
AND cont4 1

Bay_A1

Edit Object actAddModule
Property act4
Property actMapPrgrs
Assign
Prop: Pfrm
Value: start+prgrss*(end – start)/100
Ipo actIpo
Property
Prop: Pfrm
FrameProp:

Figure 15 PRGRS Msg pipeline

Now, the PRGRS is connected to a script :ReadProgressMsg.py which simply shown in Figure 15 below. Hence we don't need here to set the prgrss property to 0 through the property actuator (act 4) as it was the case in both the ASSGN and SHIP msgs since here there is a progress value calculated.

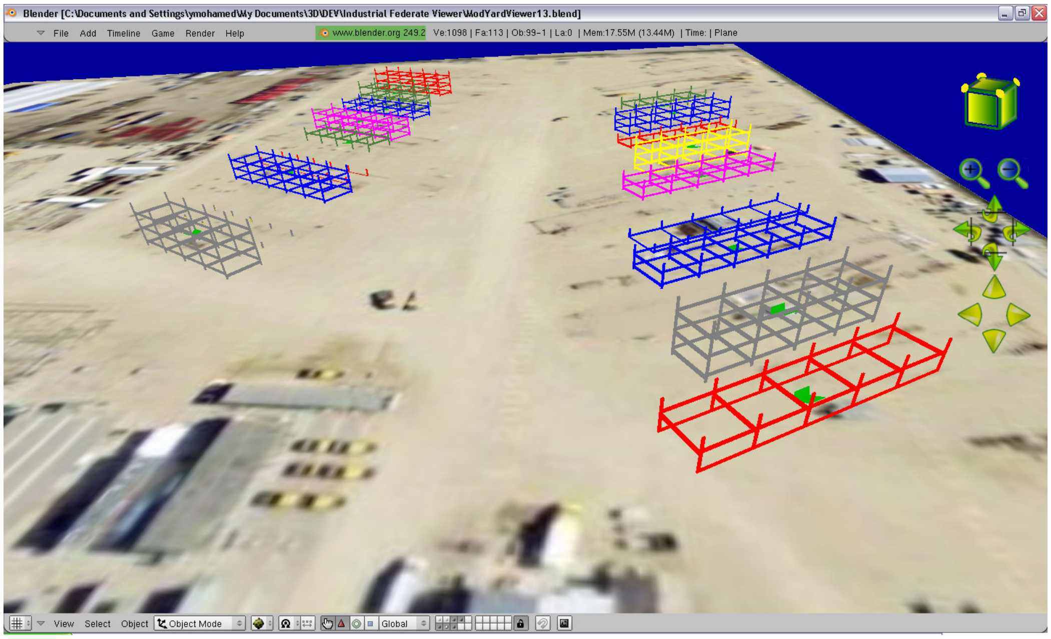


```
cnt=GameLogic.getCurrentController()
me=cnt.owner

# get msg snsor attached to controller

msgsnsr = cnt.sensors["PrgrsMsg"]

# fire only once on positive pulse
if msgsnsr.positive:

    # get the body of the message. It should have the progress % (0 to 100 range)
    # convert it to floating point number
    # make prgrss property equal to it
    # fire all the attached actuators

    progress = float(msgsnsr.bodies[0])

    if progress <= 100:
        print progress
        me['prgrss']=progress
        for act in cnt.actuators:
            cnt.activate(act)
```

Figure 16 ReadProgressMsg.py script

This script simply sets the ***prgrss property*** of the module and its parent bay object to be equal to the value that is included in the PRGRS msg body through the ; progress = float(msgsnsr.bodies[0]) ; and it is then connected to the Pfrm Property actuator (actMapPrgrs) to calculate the value of the Pfrm (frame number when the animation should stop). Then it connects to the IPO actuator to trigger the animation.

That is currently how it looks like when building:


Blender [C:\Documents and Settings\ymohamed\My Documents\3D\DEV\Industrial Federate Viewer\ModYardViewer13.blend]
File
Add
Timeline
Game
Render
Help
www.blender.org 249.2
View
Select
Object
Object Mode
Global

Ve:1098 | Fa:113 | Ob:99-1 | La:0 | Mem:17.55M (13.44M) | Time: | Plane

Figure 17 Final look at the building blocks inside the Yard simulation visualization

Appendix IV: Mesh Generation Mechanism Python Code

This appendix shows how the mesh generation python code which was explained in Chapter 4. The code is developed inside the site visualization BGE, and its coding language is Python which is an open source language.

Code:

```
# means a comment

# Path Space Module

# Defines the available space for path nodes


# 1- Must be initialized with:

#          - Whether we are dealing with 2D or 3D space (default to 2D)

#          - Overall space limits in global world coordinates

#             Centre, X, Y, Z limits (if 3D)

#             defaults (center[0,0,0], 30 for Max X,Y,Z

#          - Number of nodes in each axis direction (default 10)

#             this defines the level of resolution required and affects speed of calculations

# 2- Populate the SpaceMesh variable with a complete grid

# 3- Subtract nodes from the space mesh that lie inside objects in the model

# 4- Update nodemesh variable with the reduced space mesh

# 5- Store it in a global variable to make it accissible outside the module

#==============================================================
============


import Blender

from Blender import *

import GameLogic as gl
```

```
#me=gl.getCurrentController().owner

ob=gl.getCurrentScene().objects["OBLogicBrick"]


# Initialization flag

Intit=0

path=None


# number of dimensions (2D/3D)

ND=2


# Center and X, Y, Z limits

Center=[150,150,0]

LimitX=300

LimitY=300

LimitZ=60


# number of nodes in each direction

NX= 30

NY= 20

NZ= 10


nodemesh=[]


def generateSpace():


        SpaceMesh=[]
```

```
XMin=Center[0]-LimitX/2
XStep=LimitX/NX
XMax=Center[0]+LimitX/2


YMin=Center[1]-LimitY/2
YStep=LimitY/NY
YMax=Center[1]+LimitY/2


ZMin=Center[2]-LimitZ/2
ZStep=LimitZ/NZ
ZMax=Center[2]+LimitZ/2


for x in range(XMin,XMax+XStep,XStep):
        for y in range(YMin,YMax+YStep,YStep):
                adjacent=[]
                if ND==2 :
                        z=Center[2]
                        node=[x,y,z]
                        fr=[x,y+YStep,z]
                        bk=[x,y-YStep,z]
                        rt=[x+XStep,y,z]
                        lt=[x-XStep,y,z]
```

```
# normal nodes
around=[lt,rt,bk,fr]

# remove adjacents to special boundary nodes
i=4
if x==XMin:
    around.pop(i-4)
    i-=1
if x==XMax:
    around.pop(i-3)
    i-=1
if y==YMin:
    around.pop(i-2)
    i-=1
if y==YMax:
    around.pop(i-1)
    i-=1

for a in around:
    hit,p,n=ob.rayCast(a,node,0,"collision",0,1)
    if not hit:
        adjacent.append(a)

if len(adjacent) >1:

    nodeInfo=[node,adjacent]
    SpaceMesh.append(nodeInfo)
```

```
                                        #print nodeInfo
                              else:
                                        # 3D (to do)
                                        pass
          return SpaceMesh


# Generate nodemesh (the clear path space)
def update():
          global nodemesh
          nodemesh = generateSpace()


update()
#print nodemesh
# Remove nodes that lies inside objects in the scene from the path space
# Store the remaining clear path space in the node mesh
```

Appendix V: A* Algorithm Python Code

This appendix shows how the A* Algorithm python code which was explained in Chapter 4. The code is developed inside the site visualization BGE, and its coding language is Python which is an open source language.

Code:

```
# means a comment

# This contains the PATHFINDER class,

# which contains classes for NODE and PATH.

# It's for A* pathfinding.

INIT = 0


class PATHFINDER:

    import math


    class NODE:

        def __init__(self, PATHFINDER, position, adjacentpos):

            self.PATHFINDER = PATHFINDER


            self.position = position

            self.adjacentpos = adjacentpos

            self.parent = None

            self.adjacent = []


            self.G = 0

            self.H = 0

            self.F = 0
```

```
def getG(self):
    PATHFINDER = self.PATHFINDER

    node = self
    G = 0.0
    run = 1
    while run:
        if node.parent:
            G += PATHFINDER.getRealDistance(node.position, node.parent.position)
            node = node.parent
        else:
            run = 0
    return G

def evaluate(self, targetNode):
    PATHFINDER = self.PATHFINDER

    G = self.getG()
    H = PATHFINDER.getRealDistance(self.position, targetNode.position)
    F = G+H

    self.G = G
    self.H = H
    self.F = F

    return F, G, H
    def fastEvaluate(self, startNode, targetNode):
        PATHFINDER = self.PATHFINDER
        G = PATHFINDER.getRealDistance(startNode.position, self.position)
        H = PATHFINDER.getRealDistance(self.position, targetNode.position)
        F = G+H

        self.G = G
        self.H = H
        self.F = F

        return F, G, H



class PATH:
    def __init__(self, startpos, targetpos, startNode, targetNode, OPEN, CLOSED):
        self.OPEN = OPEN
        self.CLOSED = CLOSED

        self.startpos = startpos
        self.targetpos = targetpos

        self.nodes = []

        self.nodes.append(targetNode)
        node = targetNode
        run = 1
        while run == 1:
            if node.parent:
                node = node.parent
                self.nodes.append(node)
            else:
                run = 0

        self.nodes.reverse()

        self.path = self.getPath()

    def getPath(self):
        path = []
        path.append(self.startpos)
        for node in self.nodes:
            path.append(node.position)
        path.append(self.targetpos)
        return path
   #GETS THE TRAVELLEDDISTANCE

# Compiles the adjacents for each node in a list.
def compileAdjacents(self, nodes):
    for node in nodes:
        adjacent = []
        for targetnode in nodes:
            if targetnode.position in node.adjacentpos:
                adjacent.append(targetnode)
        node.adjacent = adjacent

# Takes in a vertinfolist and converts it into a list of node objects.
def NodemeshToNodes(self, nodemesh):
    nodes = []
    for vertinfo in nodemesh:
        position = vertinfo[0]
        adjacentpos = vertinfo[1]
        node = self.NODE(self, position, adjacentpos)
        nodes.append(node)
    self.compileAdjacents(nodes)
    return nodes

def makeNodes(self, nodelist):
    nodemesh = nodelist
    nodes = self.NodemeshToNodes(nodemesh)
    return nodes



def cleanNodes(self, nodes):
    for node in nodes:
        node.parent = None
        node.G = 0
        node.H = 0
        node.F = 0
# Gets the distance between positions A and B.
def getRealDistance(self, A, B):
    math = self.math
    X = abs(A[0] - B[0])
    Y = abs(A[1] - B[1])
    Z = abs(A[2] - B[2])
    Ds = (X*X)+(Y*Y)+(Z*Z)
    D = math.sqrt(Ds)
    return D

def getManhattanDistance(self, A, B):
    math = self.math
    X = abs(A[0] - B[0])
    Y = abs(A[1] - B[1])
    Z = abs(A[2] - B[2])
    D = X+Y+Z
    return D

# Returns the nearest node, given a position and a list of nodes.
def getNearestNode(self, position, nodes):
    best = []
    for node in nodes:
        if best:
            bestnode = best[0]
```

```
                bestdistance = best[1]
                distance = self.getRealDistance(position, node.position)
                if distance < bestdistance:
                    best = [node, distance]
            else:
                distance = self.getRealDistance(position, node.position)
                best = [node, distance]
        return best[0]


    # Returns the nearest visible node, given a position, list of nodes, and a gameobj (for
raycasting)
    # btw, this is pretty expensive, it does a lot of raycasting (to each node).
    def getNearestVisibleNode(self, position, nodes, gameobj):
        best = []
        for node in nodes:
            if best:
                bestnode = best[0]
                bestdistance = best[1]

                obj, point, normal = gameobj.rayCast(node.position, position)
                # If the position can see the node...
                if not obj:
                    # And the distance is less then the current best's distance...
                    distance = self.getRealDistance(position, node.position)
                    if distance < bestdistance:
                        # Then this node is the new bestest node!
                        best = [node, distance]
        # If there is no current best
        else:
            obj, point, normal = gameobj.rayCast(node.position, position)
            # If the position can see the node...
            if not obj:
                # Then this node is our best so far.
                distance = self.getRealDistance(position, node.position)
                best = [node, distance]

    # This must mean there are no visible nodes!?
    if not best:
        print "No visible nodes!"
        # We'll just go with the nearest one then.
        best = [self.getNearestNode(position, nodes), None]

    return best[0]



# Returns the node with the best F score: doesn't evaluate F.
def getBestF(self, nodes):
    best = []
    for node in nodes:
        if best:
            bestnode = best[0]
            bestF = best[1]
            if node.F < bestF:
                best = [node, node.F]
        else:
            best = [node, node.F]
    return best[0]
 ############
#findPath

# Given a list of nodes, start position, target position, gameobject,
# and max number of steps, this method will return a path object.
# This will find a more accurate path, but at the expense of computation speed.
def findPath(self, nodes, start, target, gameobj, steps=500):
    self.cleanNodes(nodes)

    PATHFOUND = 0

    startNode = self.getNearestVisibleNode(start, nodes, gameobj)
    targetNode = self.getNearestVisibleNode(target, nodes, gameobj)

    OPEN = [startNode]
    CLOSED = []

    for i in range(steps):

        # Get node in OPEN with lowest F
        currentNode = self.getBestF(OPEN)
```

```
# Switch it to CLOSED
OPEN.remove(currentNode)
CLOSED.append(currentNode)

# For each adjacent node
adjacentNodes = currentNode.adjacent
for node in adjacentNodes:

    # If it's not in CLOSED
    if not (node in CLOSED):
        # Get F Score
        origParent = node.parent
        node.parent = currentNode
        F, G, H = node.evaluate(targetNode)

        if not (node in OPEN):
            node.parent = currentNode
            OPEN.append(node)
        else:
            if node.G < currentNode.G:
                # Already in the OPEN list, revert to origParent
                node.parent = origParent
            else:
                node.parent = currentNode
        # Stop when we found the target node or if there are no more open nodes.
        if targetNode in OPEN:
            targetNode.parent = currentNode
            PATHFOUND = 1
            break
        if not OPEN:
            break

    if PATHFOUND:
        path = self.PATH(start, target, startNode, targetNode, OPEN, CLOSED)
        return path
    else:
        return 0

###############
#fastFindPath

# Given a list of nodes, start position, target position, and
# max number of steps, this method will return a path object.
# This will usually find a path faster, but at the expense of accuracy.
def fastFindPath(self, nodes, start, target, steps=500):
    self.cleanNodes(nodes)

    PATHFOUND = 0

    startNode = self.getNearestNode(start, nodes)
```

```
targetNode = self.getNearestNode(target, nodes)

OPEN = [startNode]
CLOSED = []

for i in range(steps):

    # Get node in OPEN with lowest F
    currentNode = self.getBestF(OPEN)

    # Switch it to CLOSED
    OPEN.remove(currentNode)
    CLOSED.append(currentNode)

    # For each adjacent node
    adjacentNodes = currentNode.adjacent
    for node in adjacentNodes:

        # If it's not in CLOSED
        if not (node in CLOSED):
            # Get F Score
            F, G, H = node.fastEvaluate(startNode, targetNode)

            if not (node in OPEN):
                node.parent = currentNode
                OPEN.append(node)
```

```
          else:
            if node.G < currentNode.G:
              # No change to parent, and already in the OPEN list.
              pass
            else:
              node.parent = currentNode

      # Stop when we found the target node or if there are no more open nodes.
      if targetNode in OPEN:
        targetNode.parent = currentNode
        PATHFOUND = 1
        break
      if not OPEN:
        break
 if PATHFOUND:
      path = self.PATH(start, target, startNode, targetNode, OPEN, CLOSED)
      return path
    else:
      return 0
Pathfinder = PATHFINDER
```