Generalized Experience Management

by

David James Thue

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Computer-based interactive environments present a compelling platform for research in Artificial Intelligence. Using games as its domains, this work has traditionally focused on building AI agents that can play games well (e.g., Checkers, Go, or StarCraft). In more recent years, a parallel line of research has aimed to achieve a different goal: to mimic the abilities of human game designers, extending their reach into the run time of the game. By building an AI agent to gather new information and make decisions as their proxy, designers can ensure that their goals are pursued in a way that adapts to each player automatically, while the game is underway.

In this dissertation, I present the Generalized Experience Management (GEM) framework, the first mathematical formalization of modifying the dynamics of an interactive environment during end-user play. Moving beyond traditional, *ad hoc* methods for creating AI agents that manage player experiences, GEM is grounded in the theory of Markov Decision Processes while still remaining practically applicable in both industry and academia. To evaluate the framework and demonstrate its versatility, I present four adaptive systems as instances thereof: two that I designed and tested through controlled user studies, one that was created independently in a commercial video game, and one that was seminal in the domain of Interactive Drama. Finally, I propose and demonstrate a detailed method for evaluating GEM systems, including a new way to distinguish between the effects of player-specific and player-independent adaptation.

Preface

This thesis is an original work by David Thue. The user studies described in this thesis received research ethics approval from the University of Alberta Research Ethics Board under two projects: (1) “Evaluating entertainment of recurring characters in adaptive interactive stories”, No. 1844 (CLG08-08-05), approved August 28, 2008, and (2) “Player Agency in Interactive Stories”, No. Pro00017135, approved January 6, 2011.

Acknowledgements

Contributors

While I led and organized the work that I present herein, much of it was done in collaboration with others. My regular discussions with Vadim Bulitko both inspired and clarified many of the ideas that form the basis of this work, and Marcia Spetch provided invaluable guidance and support for the empirical evaluations that I performed. My discussions with Howard J. Hamilton and his research group at the University of Regina also had a formative effect on the later part of my work. Eric Wasylishen, Michael Webb, Trevon Romanuik, and Charles Crittenden were instrumental to the development of my two empirical testbeds, into which they all invested long hours and incredible amounts of energy. To all of these people: Thank you. I value our collaboration immensely, and this work would have been greatly lessened without your thoughtful help.

General Thanks

My family and Lauren, for their unending and unmatched support of everything that I do –
– even when that involves moving away to live and work in Iceland.
Vadim, for his sound advice, understanding patience, and thoughtful supervision.
Marcia, for her reliable expertise, insightful comments,
and willingness to support our access to the Research Participation Pool.
Howard, for his carefully formed feedback and relentless support.
Eric, Mike, Trevon, and Charlie, for their tireless work
in bringing *Annara's Tale* and *Lord of the Borderlands* to life.
Vadim's research group, IRCL, for their helpful feedback and support of my work.
Howard's research group, for engaging discussions and distractions alike.
Duane Szafron and Jonathan Schaeffer, for guiding this work
and serving on my supervisory committee.
Osmar Zaiane and Arnav Jhala, for serving on my examining committee.
Andrei Volodin, for his knowledge of statistics and the joy with which he shares it.
Sarah Beck, Andrea Budac, Teri Drummond,
for testing *Lord of the Borderlands* before the user studies began.
Sharon Randon and Tom Johnson,
for their selfless work in support of the Research Participation Pool.
All of the participants who helped to evaluate my work.
NSERC and iCORE, for funding my research.
Reykjavik University, for taking me in and keeping their faith
that we would see this dissertation done.

Table of Contents

List of Tables

List of Figures

Glossary

Abstract States or Actions: states or actions that omit unnecessary details about the underlying interactive environment, 10

Agency: an agent's ability to influence the course of its experience in an interactive environment by performing actions therein, 6

Baseline Manager: the basis of comparison when evaluating one manager by comparing it to another, 73

Complete Trajectory: a trajectory that begins at time step 0 and contains exactly one game ending, 72

Decision Constraint Function, Balanced: a decision constraint function such that, regardless of the player's actions, (i) the same number of decision points occur for every player, (ii.a) all of the decision points that can be reached first during gameplay (i.e., before any other decision points) yield the same set of transition functions, and (ii.b) the same is true for every set of decision points that can be reached second, third, etc. thereafter, 74

Decision Constraint Function: a function that maps from player histories to non-empty subsets of the set of all transition functions (GEM Building Block #1), 25

Decision Constraints: the set of transition functions given by a decision constraint function, 25

Decision Point: any state/action pair after which more than one transition function is available for selection following some history that ends with that pair, 26

Designer: a person who decides which effect a given environment should have on each player that engages with it, 1

Environment: surroundings of an agent that change when it performs an action., 1

Estimated Player Policy: a function that maps from a given history, a state, and an action to an estimate of the probability that the player will perform the given action in the given state, given that the given history has occurred earlier in the game (GEMBuilding Block #4), 31

Expected Score: the expected value of the score that would arise from a given manager managing the experience of any player in a given population, in a given interactive experience, 76

Experience Management: the process of optimizing a player's experience in an interactive environment by changing that environment while the experience is underway, 6

Experience Manager: an agent outside of an MDP that: (a) observes all of the states and actions that occur therein, and (b) changes the MDP's transition function while any player's experience in the MDP is underway, toward maximizing that player's return, 10

Feature Vector: a vector of features, 32

Feature: a function that maps from a given trajectory to a real number, used to represent some factor of the player's experience that affects how successful it might be, 32

First Complete Trajectory: the first complete trajectory through a given interactive environment that is experienced by a given player, 76

Fixed Manager: a player-independent manager with decision constraints such that only one transition function is available to select following every state/action pair, 73

Game Ending: a state from which any player action will only ever result in the same state occurring again, 72

History: a trajectory that begins at time step 0 and ends with the most recent action that occurred, 11

Ideal Manager: a manager that identifies the set of transition function sequences for each player that would cause them to give the highest score and selects transition functions according to any sequence in that set, 71

Intended Effect: a designer-desired effect on the player that a manager attempts to achieve through its operation (*see: Designer*), 1

Interactive Environment: an environment in which the agent can perform at least two different actions, 5

Interactive Story: an interactive environment that allows players to influence the progression of a story as it unfolds, 17

Manager's Policy: a function that maps from a given interactive environment, a player's history, and optional other parameters to the single transition function that it selects for that environment to use at the next time step, 12

Manager's Score: the average of many scores obtained from a group of players, 70

Manager: (*see: Experience Manager*), 10

Objective Function: a function that maps from a transition function, a history, and the current interactive environment to a real number that estimates the player's return (GEM Building Block #2), 27

Optimal Experience: an experience such that no other experience in the same environment is more successful at affecting that player in the intended way (*see: Intended Effect*), 1

Player-Independent Manager: a manager whose opportunities to select transition functions do not depend on the player's actions, 73

Player-Specific Manager: a manager whose opportunities to select transition functions depend on the player's actions, 73

Policy: a mapping from every state in an environment to an action that the agent will perform in that state, 10

Proportional Random Variant (of some target manager): a player-independent manager that gives sequences of transition function in the proportions that were observed for a target manager, 75

Random Manager: a player-independent manager that samples randomly from a given probability distribution over transition functions at each decision point that occurs in the game, 73

Return: a function of the rewards that an agent receives receive during the course of any experience that it has in a given interactive environment; in GEM, it represents how successfully that player was affected in the way that the designer intended, 11

Rollout Function: a function that maps from a decision point, a transition function, and an interactive environment to a set of trajectories that could potentially follow after the given point occurs and the given transition function is selected by the manager (GEM Building Block #3), 29

Rollout: any trajectory in the set given by the rollout function, 29

Score Function: a function that gives a player's score for some given trajectory; used to represent a player for evaluation purposes, 71

Score: a measured value of how successfully a given player was affected by a trajectory that they experienced in a given game, 70

Sequence of Transition Functions: the ordered sequence of transition functions that a manager selected during some player's experience in a given interactive environment, 71

Target Manager: the manager of interest when evaluating one manager by comparing it to another, 73

Trajectory: a rotating sequence of states, actions, and transition functions that occur during an agent's experience in an interactive environment, 11

Uniform Random Manager: a manager that gives sequences of transition functions randomly, as drawn from a uniform distribution over the set of all possible sequences of transition functions for a given interactive environment, 80

Chapter 1

Introduction

Much research in Artificial Intelligence (AI) has focused on the task of optimizing the experience of a given single agent. Traditionally, this task has been assigned to the agent itself: it must improve its ability to choose between actions in a given environment, toward causing its own experience to achieve some designer-specified goal (e.g., finding a shortest path, or collecting the most rewards). The effectiveness of such an agent at run-time depends heavily on its ability to correctly estimate which action is the most likely to lead to achieving the goal. What should be done, however, when an agent is unable to form reliable estimates? In this work, I study the challenge of optimizing the experience of an agent that is incapable of doing so itself. In my solution, I shift the optimization problem over to a second, more capable agent (called a *manager* (Mateas 2002; Riedl et al. 2008)) that monitors and modifies the first agent's experience while it is underway. I refer to the first agent generally as a *player* henceforth.

Manager/player relationships are common in daily life, and in every such instance, the manager's goal can be expressed as a particular, *intended effect* on the player. This effect might be multi-dimensional, and the manager might wish for it to occur either during a given experience or after it has ended. For example, adults manage the experiences of young children to ensure their health and safety, teachers manage the experiences of students to encourage them to learn, and medical doctors manage the experiences of patients to maintain and improve their health. For a given *environment* (i.e., a context in which player experiences can occur), some of the experiences that are possible therein might be less successful than others at affecting the player as desired. For example, given two potential medical treatments, one might be less well suited to a given patient's lifestyle. From the perspective of the manager, an experience is *optimal* for a given player if no other experience is more successful at affecting that player in the intended way (e.g., a teacher might define their student's experience to be optimal if the student remains engaged throughout a lesson and learns a given skill well enough to perform it easily afterward). A manager tries to dynamically optimize each player's experience by changing the environment: it first estimates how successful each of several potential changes might be at affecting the player as desired, and then it carries out the change that maximizes its estimated success.

When implemented as an AI system, a manager's goal comes from a *designer* – a person who decides which effect a given environment should have on each player that engages with it. Any AI manager can thus be thought of as a software system that dynamically pursues a designer's goals, where those goals describe the intended experience of each player in a particular environment.

In general, optimizing a player's experience is a challenging problem to solve. Since interrupting the player's experience (e.g., by pausing to administer a questionnaire) could prevent them from being affected as desired, it is difficult for the manager to gather player feedback while the experience is underway. Furthermore, without a regular source of clear feedback regarding its decisions (like the reward signal in Reinforcement Learning (RL) (Sutton and Barto 1998)), the manager has no reliable way to learn the value of choosing one update to the environment over another. Additionally, given that the likelihood of a player returning for a second experience may depend heavily on the success of the first one, it is important for the manager to succeed during every player's first experience in a given environment. As a result, the manager may have no chance to improve its decisions over multiple iterations (as RL agents commonly do) before being required to perform well. Finally, to preserve the generality of this work, I assume that players may not be aware that their experience is being managed; thus, their cooperation cannot be guaranteed.

Optimizing a player's experience using an AI system is an even more challenging problem. As the variety of experiences that players can have in a given environment increases, the task of specifying which changes to the environment will be optimal following each partial experience becomes increasingly large. Furthermore, although several AI managers have been created to-date (Weyhrauch 1997; Mateas and Stern 2005; Aylett et al. 2005; Riedl and Stern 2006; Magerko 2006; Barber and Kudenko 2007; Thue et al. 2007a; Valve Corporation 2008; Arinbjarnar and Kudenko 2008; Figueiredo et al. 2008; Tomaszewski and Binsted 2009; Swartjes and Theune 2009; Thue et al. 2011; Sullivan 2012; Yu and Riedl 2012; Ramirez and Bulitko 2014), the designs of such systems have arisen most often from particular, specific domains (e.g., dramatic storytelling, computer games, or education and training). As a result, the terminology for shared concepts is unstandardized, and communicating the details of one manager to the creators of another often requires a substantial initial negotiation to find some useful common ground (Barber et al. 2008; Koenitz et al. 2010; Szilas et al. 2011; Koenitz et al. 2011; Szilas et al. 2012). Without a general, conceptual framework in which to understand the operation and components of existing AI managers, it is difficult to extend the capabilities of one manager using techniques that were designed for another.

In this dissertation, I present Generalized Experience Management (GEM): a formal specification of experience management that unifies previously disparate approaches into a common mathematical framework. When implemented, this framework produces an experience manager that automatically performs three tasks; it: (i) gathers information about an experience's current player, (ii) uses that information to assess different possible changes to a given environment, and (iii) modifies the environment toward optimizing the player's experience with respect to a set of intended effects (e.g., that the player should have fun, feel influential, etc.). The GEM framework is the primary contribution of this work; I outline two other contributions in the following paragraph.

The remainder of this dissertation is organized as follows. In the rest of this chapter, I give a high-level example of experience management in the context of computer video games and introduce the terms and concepts that are essential to understanding this work. In Chapter 2, I formulate the problem of experience management using a common AI formalism (Markov Decision Processes), and introduce the measures that I used to evaluate my work. Following my review of related research (Chapter 3), I present Generalized Experience Management in three parts: a foundation, a set of building blocks, and a description of how these can be combined to create a new manager or represent one that already exists (Chapter 4; Contribution 1). To evaluate the framework and demonstrate its versatility, I present four existing managers as instances thereof: two that were created independently by other researchers, and two that I created using commercial video game technology (Chapter 5). The last of these managers, PaSSAGE 2, is novel contribution of my Ph.D. research (Section 5.4; Contribution 2). In Chapter 6, I present and demonstrate a novel methodology for evaluating GEM managers (Contribution 3). I discuss the benefits and limitations of GEM and offer ideas for future work in Chapter 7, and conclude the dissertation in Chapter 8. Appendix A contains materials from the user studies that I conducted and Appendix B lists the publications that have arisen from this work.

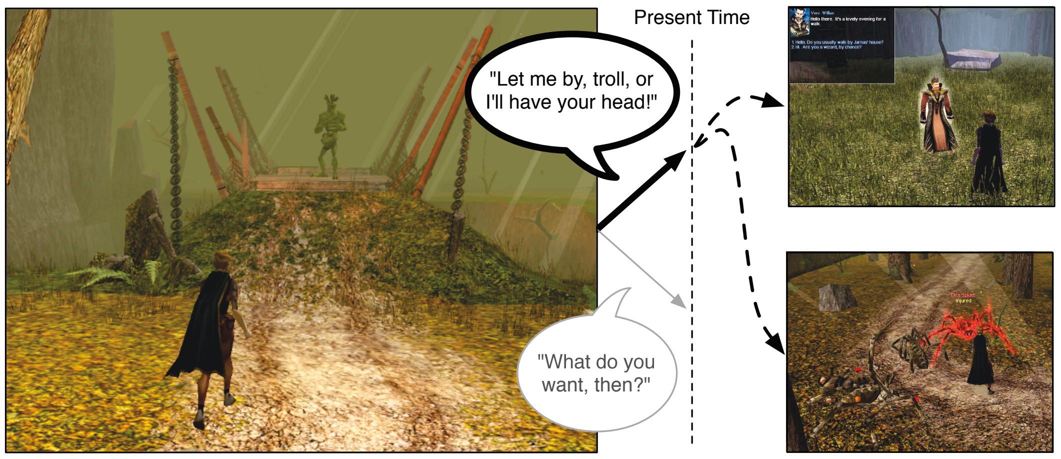

"Let me by, troll, or I'll have your head!"
"What do you want, then?"
Present Time
Vero Willian
Hello there. It's a lovely evening for a walk.
1. Hello. Do you usually walk by Jarnas' house?
2. Hi. Are you a wizard, by chance?

Figure 1.1: An opportunity to change the player's experience in *Annara's Tale*. Left pane: The player chooses to fight the troll that blocks Annara's path (thick straight arrow), and the manager can choose how this action should connect to subsequent game content (curved dashed arrows). Upper right pane: Annara meets a traveller who needs help solving a riddle. Lower right pane: Annara sees someone from her village being attacked by giant spiders.

1.1 An Illustrative Example

As an example of how an experience might be optimized by a manager, consider one of the manager decisions in *Annara's Tale*, an interactive adaptation of the story *Little Red Riding Hood* (Perrault 1697) that I created in prior work (Figure 1.1; (Thue et al. 2007a)). The player controls the story's protagonist (named "Annara", instead of "Red" from the original story), and gets sent into the forest on a mission soon after the story starts. Shortly after entering the forest, Annara meets a troll who blocks her path (Figure 1.1: Left pane). Given options to either fight or speak with the troll, suppose that the player chooses to fight (saying *"Let me by, troll, or I'll have your head!"*).

Consider the point in time right after the player starts the fight (labelled "Present Time" in the figure). Looking into the future, there are two updates to the environment that could be used to continue the story, after the fight is over. One update involves placing a traveller in the forest who has been perplexed by a riddle (upper right in the figure); he offers to help Annara in exchange for her solving it, but the player might choose to ignore him instead. The other update involves placing someone from Annara's village in the forest and having them be attacked by giant spiders (lower right); the player could fight off the spiders and obtain a reward, or they could continue along the forest path as though nothing was amiss. Back at the present time (when the player has just started to fight the troll), the manager must make a decision: of all of the available updates to the environment, which one should be carried out? By observing the player's choice to attack the troll, the manager might learn that the player is more inclined toward combat than other kinds of gameplay. Given this knowledge and the (designer-provided) goal of players having fun, the manager might choose the update that leads to even more combat (in this case, the one that leads to the spider attack).

This example is intentionally small: it covers only a single informative player action, a single manager decision, and two possibilities for what might happen next; it also does not consider how the manager's decision might influence subsequent parts of the experience. In general, a given experience could involve a wide range of informative player actions and as many manager decisions as the designer allows. Furthermore, those decisions could affect much larger sections of subsequent gameplay, limited only by the remaining available length of the experience.

1.2 Background: Interactive Environments & Experiences

In general, an *environment* is defined by three components: a set of *states* that players can perceive, a set of *actions* that players can perform, and *dynamics* that describe how player actions in each state of the environment lead to new states occurring. An *experience* occurs whenever a player both (i) perceives the different states of an environment as they change over time and (ii) performs actions therein[1]. For example, daily life is an experience because we perceive the world around us from one moment to the next and perform actions while doing so, and playing a video game is an experience

[1] I consider waiting to be an action like any other.

because we perceive the states of a virtual world via the display screen of a computer and use an input device to perform virtual actions in that world. I represent an experience as an alternating sequence of states and actions.

Whenever an environment offers at least two actions for the player to perform, that environment is said to be an *interactive environment*. For example, the simple interactive environment in Figure 1.2 Original has four states $\{A, B, C, D\}$, three player actions $\{1, 2, 3\}$, and dynamics such that action 1 in state A leads to B, action 2 in state B leads to C, and action 3 in state B leads to D.

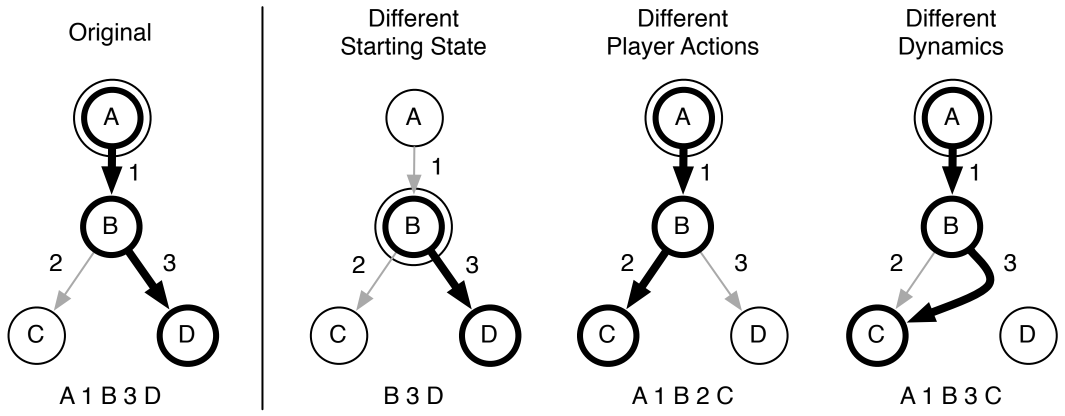


Figure 1.2: Four graphs showing how player experiences can vary under different conditions (listed across the top). Each graph is an interactive environment, and each circle is a state of the environment. Each double circle shows the player's starting state, and each arrow shows how the numbered player action leads to a new state (i.e., it shows the environment's dynamics). Thick outlines and arrows indicate the states/actions that the player perceived/performed as part of their experience, which is summarized below each graph.

In any interactive environment, the occurrence of any particular experience depends on three factors: (i) the state of the environment when the player's experience begins, (ii) the actions that the player performs, and (iii) the dynamics of the environment. Figure 1.2 shows an example of how a player's experience can vary when each of these factors is changed. Comparing the first and second graphs in the figure shows how having a different starting state causes the player to miss the first state (A) and action (1) of the original experience. Comparing the first and third graphs shows how a different player action after state B (action 2 instead of 3) causes the player's experience to end with state C instead of D. Finally, comparing the first and last graphs shows how having different environment dynamics (where action 3 in state B leads to state C instead of D) causes the player's experience to end with state C instead of D. In all four cases, the player's experience is different.

In undertaking this work, I was particularly interested in the problem of managing experiences that occur in environments where the manager can observe the states and actions that occur during the player's experience. Such managers have the opportunity to learn about each player, toward improving their ability to optimize each player's experience. For this reason, I assume that every managed experience has three properties: (i) it occurs in an interactive environment, (ii), the manager can observe the states and actions that occur therein, and (iii) the manager can influence which experience occurs (I discuss how it can do so in Section 1.3).

1.2.1 Agency & Authorial Control

Through their actions, each player exercises *agency* – the ability to influence the course of their experience in an interactive environment by performing actions therein. Feeling influential is generally considered to be beneficial in daily life (Larson 1989), but the presence of agency complicates the ways in which designers can affect players through their experiences. More specifically, while traditional, non-interactive environments (e.g., books or movies) allow designers (such as authors or film directors) to focus on affecting their audiences through a single experience that everyone will have, the agency that players have in interactive environments makes it impossible to guarantee that any *particular* experience will be had. As a result, ensuring that each player will be affected by their experience as the designer intended becomes a much larger problem. Research in Experience Management aims to solve this problem by finding ways to automatically direct each player's experience while it is underway.

1.3 Experience Management

For a manager to attempt to optimize a player's experience, it must be able to change either (i) the player's starting state, (ii) the player's actions, or (iii) the environment in which their experience occurs. Changing the starting state amounts to optimizing the player's experience before any interaction has occurred, a point at which the manager would have had no opportunity to learn from its observations of the player's actions. To focus on the potential benefits of such learning, I do not explore managers that change the player's starting state in this work. To preserve the player's autonomy, I assume that the manager cannot affect which actions the player performs. I therefore define *experience management* as the process of optimizing a player's experience in an interactive environment by changing that environment while the experience is underway. More specifically, managers work by changing the environment's *dynamics* – the ways in which player actions in any given state lead to new states occurring. This focus on dynamics distinguishes Experience Management from the related AI subfield of Procedural Content Generation (PCG) (Yannakakis and Togelius 2011); while PCG focuses on creating new states or actions to allow a greater variety of player experiences, Experience Management focuses on shaping the player's experience of existing states and actions.

1.3.1 AI Managers

In the simplest case, an AI manager can update the environment's dynamics after each player action, and it does so toward ensuring that each player's experience will affect them in the way that the designer intended. For example, the AI manager in *Façade* dynamically defines a sequence of dramatic interactions (called "beats") among two virtual characters and the player, and each beat is associated with the designer's estimate of how much tension it will add to the story (Mateas 2002; Mateas and Stern 2003). Before any player experience begins, the designer provides the manager with a desired curve of tension versus time. After each player action, the manager updates the environment (e.g., by causing the player's actions to trigger particular dramatic beats) in a way that minimizes the difference between the given curve and an estimate of the amount of tension that is currently in the story. The assumption underlying this design is that the more closely the estimated progression of tension in each player's story matches the designer's curve, the more successfully the experience will affect the player as the designer intended (e.g., players should feel catharsis).

1.3.2 Evaluation & Optimality

To accelerate progress toward creating AI experience managers, it is important to build, critically evaluate, and compare complete, working prototypes (Mateas and Stern 2003). To date, however, such efforts have only compared managers from *within* a single subfield of Experience Management research (e.g., in Drama Management or Dynamic Difficulty Adjustment). An important step toward comparing managers *across* subfields is the creation of a framework that allows managers from different subfields to be represented in a unified way. As I show in Chapter 4, GEM can be used to represent managers from the subfields of Drama Management and Dynamic Difficulty Adjustment.

Measuring the optimality of a player's experience (i.e., its success at affecting the player as intended) can be a daunting task, because measures of how much fun they had or how excited they felt are often difficult to obtain in absolute terms. Throughout this work, I assume that the manager is unable to directly observe how each of its changes to the environment affect its overall success at affecting the player. For the experimental part of this work (Chapter 6), I mitigate the problem of lacking absolute measures of optimality by comparing the *relative* success of different managers in a given interactive environment. Specifically, by demonstrating that one manager is more successful at achieving an intended effect in players than another manager (e.g., that it makes a video game more fun), one can show progress toward creating managers that can provide each player with an optimal experience (at least within that given environment).

Given the ability to perfectly predict the player's actions and to accurately measure the success of any given path through the game, an optimal manager could, in theory, be created. Until such abilities are available, however, experience managers can still *pursue* optimality in practice, even if achieving it remains out of reach.

1.4 Domain & Research Focus

Throughout this work, I frequently discuss experience management in the context of computer video games. Such games are not only excellent contexts for concrete, detailed examples of experience management, but they are also particularly useful as an evaluation domain when investigating AI managers. There are several reasons why this is so. First, (human) game designers often aim for their games to have particular effects on their players (e.g., the feelings of fun, frustration, or excitement), which makes them good models for AI managers. Second, the virtual nature of video games makes it easier to pursue the designer's goals with an AI-driven approach, since there is no need to create interfaces to robotic arms, manual switches, or other devices in the "real world"; this helps simplify the logistics of performing this kind of research. Finally, the potential benefits of applying AI experience management in video games are worth pursuing from a commercial perspective, since increasing the amount of fun that players have while playing could have a direct, positive impact on a game's reception, longevity, and sales.

Although several commercial video games include AI experience managers (e.g., "rubber-banding" in *Mario Kart 64* (Nintendo 1996), or the "AI Directors" in *Left 4 Dead* (Valve Corporation 2008) and *DarkSpore* (Maxis Software 2011)), the scope of such systems is often limited by at least one of three factors: (i) a lack of ways to change the environment (which can be expensive to create), (ii) a lack of useful information about the player (which can be difficult to acquire unobtrusively), and (iii) the large amount of labour required to assign every combination of player information to the best experience for that player. Although a lack of ways to change the environment (the so-called "Authoring Bottleneck Problem" (Csinger et al. 1994)) is a serious concern, I focus my attention on the latter two problems for the following reasons: the first factor holds the interest of several other scholars (Riedl et al. 2008; Orkin and Roy 2009; Ontañón and Zhu 2011; Swanson and Gordon 2012; Li et al. 2013) while the latter two are comparatively less explored, and there already exist several commercial video games which, while not lacking in content, might still benefit from having their players' experiences being managed (e.g., *Skyrim* (Bethesda Game Studios 2011), *Grand Theft Auto: V* (Rockstar North 2013), or *Red Dead Redemption* (Rockstar San Diego 2010)). Looking beyond the domain of video games, real-world experiences are often modifiable in a variety of ways without any authoring effort, making the authoring bottleneck problem somewhat less of a concern. For example, a parent can remove dangerous objects from a room before a child enters (thereby managing the child's safety), without any new objects needing to be created or acquired.

1.5 Thesis Statements

The thesis statements that govern this work are as follows, both of which concern Generalized Experience Management (GEM), the framework that I propose in Chapter 4 (Contribution 1).

1. *Generalized Experience Management unifies and supports the comparison of managers from previously separate subfields of experience management research (e.g., Drama Management and Dynamic Difficulty Adjustment);*

2. *Generalized Experience Management supports the creation of managers that are more successful at affecting their players as intended than a baseline competitor.*

I support the first claim by demonstrating how GEM can be used to represent four different managers: Weyhrauch's (1997) seminal Moe system (Drama Management), the AI Director (Booth 2009) from the video game *Left 4 Dead* (Dynamic Difficulty Adjustment), and two managers that I created using commercial game engine technology; PaSSAGE (Drama Management; (Thue et al. 2007a; 2010a)), and PaSSAGE 2 (Drama Management; (Thue et al. 2010b; 2011); Contribution 2). To support the second claim, I developed a new method for evaluating experience managers (Contribution 3) and then applied it in several user studies of PaSSAGE and PaSSAGE 2.

Summary

In this chapter, I introduced *experience management* as the task of optimizing a player's experience in an interactive environment, while it is underway, by changing that environment's dynamics. I defined *optimality* in terms of the success that a manager has in ensuring that each player's experience affects them in a particular intended way, as specified by a designer. I then described how optimizing a player's experience using AI systems is particularly challenging for two primary reasons: the problem itself involves ambiguous feedback, no time for training, and a potentially uncooperative player, and the process of creating such systems still suffers from a lack of standardization across the variety of current approaches. I offered a high-level example of how experience management might proceed, and then explained how *agency* complicates the problem of ensuring that each player's experience affects them in the intended way. I discussed the challenge of evaluating managers in the face of subjective designer goals, which I approach by comparing the *relative* success of different competing managers. I also emphasized that progress in this area of research can be accelerated by comparing existing managers – not only within a given subfield, but across subfields as well. Finally, I explained how video games offer a viable domain for studying experience management, because game designers often express their goals in terms of intended effects on players (e.g., having fun), virtual worlds offer simplified research logistics, and managing player experiences in video games might improve their commercial value.

Chapter 2

Problem Formulation

In this chapter, I formally define the task of experience management in terms of Markov Decision Processes (Bellman 1957) and state the criteria for success of this work.

2.1 Experience Management with Markov Decision Processes

A Markov Decision Process (MDP) is a discrete-time, stochastic control process wherein an agent alternately perceives *states* ($s \in S$) and performs *actions* ($a \in A$) that can influence which state occurs next in the process. Both the states and the actions of an an MDP might be *abstract*, meaning that they might omit unnecessary details about the underlying interactive environment (e.g., the colour of a video game character's hair). The process starts with time step $t = 0$, and t increases by one immediately before each new state occurs. A *transition function* (τ) governs the occurrence of subsequent states ($\tau : S \times A \times S \to [0, 1]$), where $\tau(s, a, s')$ gives the probability of state s' occurring after the agent performs action a in state s. Each time the agent performs an action, they receive a scalar *reward* ($\mathcal{R} : S \times A \times S \to \mathbb{R}$) and a new state occurs as determined by the transition function. I use subscripts to denote states and actions that occur at a given time step t (e.g., s_t or a_t).

Traditionally, MDPs have been used in Artificial Intelligence (AI) research to investigate AI systems that act as agents therein (e.g., Reinforcement Learning (Sutton and Barto 1998)). Such systems often attempt to create a *policy* (i.e., a mapping from every state in S to some action in A that the agent will perform in that state) that maximizes some function of the rewards that the agent receives over the course of their experience; the value given by this function is called the agent's *return*. In the context of experience management, the similarity between the dynamics of an environment (recall Section 1.2) and the transition function of an MDP suggests that the *player* can be thought of as an agent in an MDP, with their return serving as a measure of how successfully they were affected by their experience therein. Under this formulation, a *manager* is an agent outside of an MDP that: (a) observes all of the states and actions that occur therein, and (b) changes the MDP's transition function while any player's experience in the MDP is underway, toward maximizing that player's return. It must do so, however, without any ability to control or compute the functions that give the player's rewards or the player's return. To the best of my knowledge, this representation of a manager is a novel contribution, and it helps to unify previous research in both Drama Management and Dynamic Difficulty Adjustment (as I explain in Chapter 5). To simplify my writing henceforth, I refer to MDPs as "games" and player experiences therein as "gameplay" without loss of generality.

2.1.1 Interactive Environments, Trajectories, and Histories

Using MDP terminology, an *interactive environment* is a tuple $e = \langle S, A, (\tau_0, \dots) \rangle$ where S is a set of states, A is a set of actions, and the sequence of transition functions $(\tau_0, \dots)$ shows that the environment might be *non-stationary* – that is, its transition function might change from one time step to the next[1]. If the environment's transition function always remains constant (i.e., $e = \langle S, A, (\tau, \tau, \dots, \tau) \rangle$ for some fixed transition function τ), then I say that it is *stationary* and use the abbreviated form $e = \langle S, A, \tau \rangle$ to denote it.

Let $\mathcal{T}$ be the set of all possible transition functions that could be defined for S and A. For every action a_t the player performs during gameplay, the manager receives s_t and a_t as observations and chooses a transition function τ in $\mathcal{T}$ toward maximizing the player's return (I describe the return in more detail in Section 2.1.2). Once τ has been selected, the manager sets the environment's transition function to τ (i.e., $\tau_{t+1} = \tau$) and the environment uses it to generate s_{t+1} (by sampling from the probability distribution over S that is defined by $\tau(s_t, a_t, \cdot)$). I use $e_t = \langle S, A, \tau_t \rangle$ to denote the configuration of e at time step t, where τ_t is the transition function that is used to generate s_t. While e might be non-stationary over the course of any player's experience therein (i.e., over the course of multiple time steps), it is effectively stationary at every *specific* time step t that occurs (i.e., every $e_t = \langle S, A, \tau_t \rangle$ is stationary even if $e = \langle S, A, (\tau_0, \dots) \rangle$ is not).

Managing a player's experience produces a rotating sequence of states, actions, and transition functions that I call a *trajectory*. For example, when examined from time step $t = 0$ onward, a trajectory takes the form: $(\tau_0, s_0, a_0, \tau_1, s_1, a_1, \tau_2, s_2, \dots)$. Formally, a trajectory $h \in H$ is a sequence of the form $(\tau, s, a, \tau', s', a', \dots)$ such that for every general subsequence (s, a, τ', s') in h, the probability given by $\tau'(s, a, s')$ is greater than zero (i.e., h could be experienced by some player of the game). A *history* (h_t, at time t) is any trajectory that starts from time step 0 and ends with action a_t (e.g., $(\tau_0, s_0, a_0, \dots, \tau_{t-1}, s_{t-1}, a_{t-1}, \tau_t, s_t, a_t)$). For notational convenience, I use h_{-1} to represent an initial, empty history.

2.1.2 The Player's Return

I define the player's *return* as a function of the rewards that they receive during the course of any particular experience in a given interactive environment. Formally, the player receives a scalar reward following each action a_t that they perform in the environment, as given by the reward function (i.e., $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$); I assume that the manager is unable to observe these rewards. Over the

[1]In other work, an MDP is defined to be non-stationary if either its reward function (Sutton and Barto 1998) or its transition function (da Silva et al. 2006) can change over time. In this work, I consider only the latter case.

course of a player's experience in the environment, they accumulate a sequence of rewards whose length is equal to the number of time steps that have passed; for notational convenience, let the vector $\boldsymbol{r_t}$ represent this sequence of scalars at time step t (i.e., $\boldsymbol{r_t} = \langle r_1, r_2, \ldots, r_t \rangle$). I define the player's *return* (ξ) as a function that maps from any $n+1$ step history through a given interactive environment[2] (h_n) and the n rewards that the player received therein ($\boldsymbol{r_n}$) to a real number representing how successfully that player was affected in the way that the designer intended ($\xi : H \times \mathbb{R}^n \to \mathbb{R}$). In Reinforcement Learning, for example, the return is often defined as the sum of all of the accumulated rewards: $\xi(h_n, \boldsymbol{r_n}) = \sum_{i=1}^{n} r_i$ (Sutton and Barto 1998).

2.1.3 A Formal Definition of Experience Management

The task of experience management can now be formalized: given an interactive environment and the ability to observe the states and player actions that occur therein, *experience management* is the task of choosing, for each of the $n+1$ actions that the player performs ($a_t : t \in [0, n]$), the transition function $\tau \in \mathcal{T}$ that will maximize the player's return when used to determine the next state[3] (Equation 2.1), and then changing the environment's transition function to the one that was selected (Equation 2.2).

$$\tau_{t+1} = \operatorname*{arg\,max}_{\tau \in \mathcal{T}} \xi(h_n, \boldsymbol{r_n}) \tag{2.1}$$

$$e_{t+1} = \langle S, A, \tau_{t+1} \rangle \tag{2.2}$$

Since both ξ and $\boldsymbol{r_n}$ remain unknown and the number of possible transition functions ($|\mathcal{T}|$) is infinite, the right side of Equation 2.1 cannot be computed. As I discuss in Chapter 4, solutions to this problem commonly rely on either constraining the set of transition functions that are available to the manager via designer-defined rules (Section 4.1.1: A Decision Constraint Function), estimating the player's return algorithmically using a designer-defined function (Section 4.1.2: An Objective Function), or both.

Manager & Manager Policy

The way that a manager selects transition functions can be described by its *policy* (χ), a function that maps from an environment, a player history, and optional other parameters (which I denote by "…") to a transition function (Definition 2.1). As I describe in Chapter 4, these optional parameters can be functions that are used to compute the policy. I represent a manager formally as a tuple of functions $m = \langle \chi, \ldots \rangle$, as described in Definition 2.2.

Definition 2.1 (Manager Policy)**.** A manager's *policy* (χ) is a function that maps from a given interactive environment ($e_t = \langle S, A, \tau_t \rangle$), a player's history ($h_t$), and optional other parameters to the single transition function that it selects for that environment to use at the next time step ($\chi : E \times H \times \ldots \to \mathcal{T}$).

[2]The symbol h_n denotes a history that ends with the state and action that occurred at time step $t = n$ (e.g., $(\ldots, s_n, a_n)$).

[3]If more than one $\tau \in \mathcal{T}$ maximizes ξ, I assume that the $\arg\max$ operator selects between them in an equiprobable random fashion.

Definition 2.2 (Experience Manager)**.** An experience manager is a tuple $m = \langle \chi, \dots \rangle$, where χ is the manager's policy and the ellipsis (i.e., "$\dots$") represents other functions that the manager can use to compute its policy.

Algorithmic Representation

Figure 2.1 represents experience management in algorithmic form. On line 2, the algorithm observes each state of the environment that occurs and the action that the player performs therein. Line 3 maintains the player's history by updating it during each time step. Line 1 initializes the player's history to an empty sequence, which gets prepended to (τ_0, s_0, a_0) the first time that line 3 is executed (because the player's experience starts at time step $t = 0$). Line 4 uses the given manager's policy (χ) to compute a new transition function to use, and line 5 changes the environment's transition function (from τ_t) to use the one that the manager selected (τ_{t+1}). The environment then uses τ_{t+1} to determine its next state (s_{t+1}), and the time step advances when this state is presented to the user (not shown in the algorithm). Line 2 executes again as soon as the player chooses an action to perform in the most recently presented state; lines 2 to 5 execute once (in order) for each state/action pair that occurs.

```
Input:       $e_0 = \langle S, A, \tau_0 \rangle$: the interactive environment at time step 0
             $m = \langle \chi, \dots \rangle$: an experience manager
Observable:  $t$: the current time step of the environment
             $s_t$: the state of the environment at each time step
             $a_t$: the player's action at each time step
1  $h_{-1} = ()$
2  for each $\langle s_t, a_t \rangle$ that occurs in the environment
3      $h_t = (h_{t-1}, \tau_t, s_t, a_t)$
4      $\tau_{t+1} = \chi(e_t, h_t, \dots)$
5      $e_{t+1} = \langle S, A, \tau_{t+1} \rangle$
```

Figure 2.1: Pseudocode for performing experience management in a given interactive environment using a given manager. Recall that the set of all possible transition functions ($\mathcal{T}$, from which the manager ultimately selects) is defined implicitly by S and A.

2.1.4 Benefits of Changing the Transition Function

Managing player experiences in a given MDP by changing its transition function is at least as expressive as giving players a stationary MDP (with respect the experiences that players can have therein), since a manager could be defined that always selects the same transition function at every time step, and $e = \langle S, A, (\tau, \tau, \dots, \tau) \rangle$ is a stationary MDP.

The primary advantages of managing an experience by changing its transition function (instead of defining a stationary MDP) come from the fact that it distinguishes clearly between defining the way in which subsequent states are generated (i.e., "defining how the world works" via the transition function) and the ways in which this generation process can be altered (i.e., "changing how the world works" via the manager's policy). This factored representation affords flexibility: if ever there are insufficient resources to compute the manager's policy, the player's experience can continue uninterrupted by simply maintaining the current transition function until a new one can be set. It also simplifies the conceptual design of new environments by keeping their state space compact. For example, instead of conceptually duplicating every state of a video game to support an "easy" mode and a "hard" mode (thereby doubling the size of the state space), one can instead define two different transition functions over the original set of states (e.g., as alternate sets of parameters that govern gameplay) and then allow the player or the manager to choose between them during play to control the game's difficulty.

2.1.5 Implementation Complexity

To implement an AI manager, the designer must specify which of the $|\mathcal{T}|$ possible transition functions should be selected for every possible history that could occur in a given interactive environment. For simplicity (and without loss of generality in practice), suppose that the maximum length of every player experience is finite (n time steps) and that the number of possible transition functions ($|\mathcal{T}|$) is finite as well. The number of possible histories ($|H|$) is then given by Equation 2.3:

$$|H| = \sum_{i=1}^{n} (|S||A||\mathcal{T}|)^i \tag{2.3}$$

This number is enormous for even tiny domains. For example, with only 10 possible states, 5 player actions, experiences with a maximum length of 10, and a restricted number of possible transition functions such that $|\mathcal{T}| = |S||A||S|$, the number of possible histories exceeds 10^{40}. The rapid growth of Equation 2.3 necessitates using an abstraction over player histories when specifying the manager's behaviour. I discuss two methods for performing this kind of abstraction in Chapter 4: Decision Constraints in Section 4.1.1 and Features in Section 4.1.5.

2.2 Domain Restrictions

As I discussed in Chapter 1, video games offer a convenient domain for researching experience management. I adopt this domain for the examples and experimental section of this work, and further focus my attention on single-player video games. Managing a shared experience for multiple players would be an even *more* difficult task, for it could end up trying to optimize the experiences of players who are affected by the things that happen therein in drastically different ways. The fact that many successful commercial games have primarily single player experiences (e.g., *Skyrim*, *Super Mario Galaxy 2*, *Portal 2*, or *Grand Theft Auto V*) supports the relevance of my focus in this regard.

2.3 Definition of Success

In Chapter 4, I propose the Generalized Experience Management (GEM) framework: a modular, mathematical formalism for experience management. I created GEM for two reasons: to simplify comparing existing managers and to support creating new managers that can affect their players in the ways that the designer of an environment intended.

The utility of GEM as a comparative framework depends on it being simultaneously (i) general enough to represent diverse managers in a unified way and (ii) specific enough to the problem of experience management that it supports valuable comparisons between managers. In this instance, I use "valuable" to mean "useful in the context of further research or practical applications". A framework that was very general but ignored the problem of experience management (e.g., "every AI manager can be represented as a C++ program") would have little utility, because the dimensions along which managers could be compared (e.g., types of data structures or libraries used) would be too distantly related to the larger problem of managing a player's experience. Meanwhile, a framework that was highly specific to experience management but lacked generality (e.g., the technical description of a particular manager) would also have little utility, because it would be difficult to understand how a variety of managers could be represented therein. I demonstrate how GEM can be used to represent and unify diverse managers in Chapter 5, where I use it to represent managers from two subfields of experience management research: Drama Management, and Dynamic Difficulty Adjustment. In Chapter 7, I discuss how these representations also serve to demonstrate the value of GEM-guided comparisons.

To evaluate whether or not managers created using GEM can successfully affect their players as intended, it is convenient to measure a manager's performance at doing so in comparison to some baseline competitor. In Chapter 6, I discuss my application of an empirical way to do so: compare players' assessments of their experiences (via post-play questionnaires) across two sample groups. In one group, each player's experience is optimized by a GEM manager, while in the other group, each manager decision that arises during the player's experience is performed randomly[4]. After each player's assessment is converted to a numeric score, statistical analysis can be used to determine whether there was a difference between the average scores of the two sample groups, and provide a degree of confidence concerning whether or not that difference (if present) occurs in the population from which the samples were drawn. If the average score obtained for the manager is higher than that of the random baseline and the reported degree of confidence is high, one could deem that manager to affect its players more successfully than its baseline competitor. In Chapter 6, I present the results of applying this evaluation method to two GEM managers that I created in the context of two interactive environments. PaSSAGE aims to maximize player fun in *Annara's Tale* (Thue 2007; Thue et al. 2007a; 2007b; 2010b), and PaSSAGE 2 aims to strengthen players' beliefs that they have agency in *Lord of the Borderlands* (Thue et al. 2010a; 2011).

[4]I describe the precise nature of this randomization in Chapter 6.

Summary

In this chapter, I formally defined an experience manager as an agent that repeatedly changes the transition function of an *interactive environment*, which I represent as a Markov Decision Process (MDP) in which the *player* acts as an agent. Immediately following each player action, the manager uses its *policy* to choose a new transition function for the MDP to adopt, toward maximizing the player's *return* (a function of the rewards that they receive over the course of their experience). I explained the benefits of representing experience management as changing a transition function (flexibility and conceptual simplification), and demonstrated that this representation sacrifices none of the expressive power of a traditional (stationary) MDP. I described why implementing AI managers is particularly challenging, because the number of histories that must be mapped to transition functions grows rapidly with the size of the environment and the length of the experiences that players can have therein. I restricted the focus of this work to *single player* experiences, leaving multiplayer concerns for future work. Finally, I explained the two ways in which I will evaluate my work on Generalized Experience Management (GEM): 1) by using GEM to represent managers from the subfields of Drama Management and Dynamic Difficulty Adjustment, and 2) by conducting controlled user studies with post-play surveys for two GEM managers that I created.

Chapter 3

Related Work

The problem of experience management spans several areas of research, including both Drama Management and Dynamic Difficulty Adjustment. In the following sections, I describe different, pre-existing ways of formulating the problem of experience management in both of these areas, to help motivate the need for a more general, unifying framework.

3.1 Drama Management

Drama Management is a subfield of experience management research that focuses on automatically optimizing players' experiences in *interactive stories* – interactive environments that allow players to influence the progression of a story as it unfolds. Early investigations of AI for experience management in this context can be traced to work by Bates et al. on the OZ Project at Carnegie Melon University (Bates 1992). Much of this work was first synthesized by Laurel (1986; 1991) from the perspective of formal dramatic theory, which was later developed from a computational perspective by Weyhrauch (1997).

3.1.1 The Playwright

In her work, Laurel envisioned an AI system (called "the PLAYWRIGHT") that uses a designer-provided knowledge base and the history of an interactive environment to produce a "script" – that is, a temporally ordered sequence that describes the next few moments of the simulation that drives the environment. The design of Laurel's PLAYWRIGHT uses many of the components of experience managers that I discuss in Chapter 4 (e.g., the abilities to simulate potential futures or vary what happens based on observed player behaviour). However, its output is more restricted than that of the general experience managers that I defined in Chapter 2. Specifically, while the PLAYWRIGHT would produce a prescribed sequence of states that depict character actions and other occurrences in the simulation, a more general manager would produce a transition function to govern the behaviour of the entire simulation.

3.1.2 Moe

Weyhrauch developed Laurel's work on the PLAYWRIGHT by creating an AI experience manager called Moe (Weyhrauch 1997). During this work, Weyhrauch extended Laurel's formulation of a manager in two important ways. First, he identified that experience management could be framed as an optimization problem, where the manager estimates the optimality of any given story using one or more designer-provided "features" – that is, measures of some particular aspect of optimality, such as the player's level of engagement. Second, he introduced the notion that a manager might need to change not only the immediate, subsequent behaviour of the simulation, but also its behaviour at some potential future states. For example, to ensure that the player would discover a piece of evidence in an interactive detective story, Moe could assert a simulation rule of the form: "whenever the player visits the balcony, highlight the footprints in the mud below". This rule could be asserted far in advance, long before the player ever decides to visit the balcony. Moe's ability to broadly influence the transitions of a simulation is consistent with my formulation of experience management. However, Moe's specification is both verbose (consisting of programmatic scripts, diagrams, and associated documentation) and tightly integrated into its interactive detective domain, making it difficult to use its underlying framework to analyze other managers. The framework for experience management that I propose in Chapter 4 (GEM) is both concise and domain-independent. I describe Moe in more detail in Section 5.1, where I use it as a case study in demonstrating GEM.

3.1.3 Families of Story Managers

Since Weyhrauch created Moe, many other experience managers have been created using a variety of AI techniques. In some cases, a group of managers that all use the same technique can be thought of as a family that subscribes to the same underlying framework. I discuss two such families in the following sections: one that uses MDP solvers, and one that uses partial-order planners.

MDP Solvers

Following Weyhrauch's optimization-based approach, the managers created by Nelson et al. (2006) and Roberts et al. (2006) use Markov Decision Processes to estimate policies that are optimal with respect to various properties of player experiences therein. However, instead of using an MDP to describe the player's experience in an interactive environment (as I do in Chapter 2), they viewed the *manager* as an agent in an MDP that is distinct from the player's environment. They defined this MDP as follows. Each state represents a combined record of everything that the player has done in their environment and every action that the manager has taken thus far. Each action represents an opportunity for the manager to change the behaviour of the player's environment (as Moe could). The transition function represents how the player decides what to do in their environment, given a record of what they have done previously in their environment and the manager's previous actions. Finally, the MDP's reward function measures properties of given state/action sequences through the MDP, such as how often players willfully follow hints that encourage particular player behaviours. Given such an MDP, the manager's policy is determined by solving the MDP using standard techniques (i.e., each of the manager's actions will maximize the expected value of the reward function, once a solution has been found). Although this formulation of experience management allows the manager's policy to be generated by solving an MDP, it has two significant limitations. First, the solution found for any player will only be optimal when the MDP's transition function accurately represents the player's behaviour, but the formulation does nothing to ensure that an accurate representation is obtained. Second, it does not concretely specify how the manager's actions can affect the player's experience, referring only generally to the kinds of actions that Moe could perform. In addition to formally specifying the effect that any manager has on the player's experience, the framework that I propose in Chapter 4 includes an estimated player policy as one of its components, thereby addressing both limitations of the formulation used by this formulation of experience management.

More recently, Rowe and Lester (2013) proposed to represent experience management as a modular reinforcement learning problem, where each of a manager's decisions is represented by a distinct MDP. Their manager can thus be thought of as a collection of coordinated agents, each of which performs actions in an MDP that has relatively few states and actions. The initial state of each MDP represents a proposition about some subset of the state of the player's environment (e.g., "the player has asked a virtual character for their backstory" or "the player has stolen a character's wallet"). From each MDP's initial state, the actions that can be performed by the manager represent the its options for changing the environment's behaviour (e.g., determining how a virtual character should respond to the player). The transition function of each MDP represents the manager's uncertainty about how the player will behave following manager's decision (e.g., it might lead to a new state wherein the player has become afraid); this is similar to how Nelson et al. (2006) and Roberts et al. (2006) used the transition function in their MDP representation, and different from the formulation that I proposed in Chapter 2 (i.e., Rowe and Lester view the *manager* as an MDP agent – not the player). The manager's task is to learn a policy for selecting actions in each MDP, and the policies that are learned over all of the MDPs (one for each manager decision) are then combined through an arbitration procedure that handles any conflicts that might occur. For example, the player might ask for a character's backstory while also stealing their wallet, thus requiring two manager decisions that involve the same character. To allow their manager to learn the needed policies, Rowe and Lester used off-policy learning with a collection of gameplay traces and survey responses that they gathered from real players using a manager that chose actions randomly. The primary requirement of Rowe and Lester's modular approach is that all of the MDPs must be independent from one another; specifically, two MDPs are independent if their states represent only propositions about *disjoint* subsets of the states of the player's environment (Rowe and Lester 2013). For example, two MDPs cannot both have states that represent the proposition "the player has stolen a character's wallet". In comparison to the framework that I propose in this work, this requirement is limiting because it prevents the manager from conditioning its current behaviour on its prior behaviour, which limits the potential variety of its operations. By including previously selected transition functions in its representation of the player's history (Chapter 2), my formulation of experience management overcomes this limitation. Table 3.1 compares Rowe and Lester's MDP formulations to that of Nelson et al. and Roberts et al.

MDP Components	**MDP-based Approach**	
	(Nelson et al. 2006) (Roberts et al. 2006)	(Rowe and Lester 2013)
States	All prior player actions & manager actions	A proposition about the state of the player's environment
Actions	The manager's actions	
Transition Function	Determines the next state (given states & actions) and models uncertainty about player actions	
Reward Function	Measures designer-desired properties of state/action sequences (collected online)	Measures players' experiential outcomes (collected offline)

Table 3.1: A comparison of two approaches based on solving MDPs.

Partial-Order Planners

Several independent research efforts have investigated how partial order plans can be used to represent stories. A detailed description of AI-based planning is beyond the scope of this dissertation, but having a high level understanding is important for the remainder of this section. A partial-order plan is a directed, acyclic graph where nodes represent actions that could be performed in some interactive environment, and edges represent constraints on the execution order of pairs of nodes. If two nodes are connected by an edge, the action at the start of the edge must be performed prior to the action at the end of the edge (thus defining a partial order over nodes). Each action (e.g., "lift the block") is associated with a set of preconditions and a set of effects. Preconditions and effects are propositions about the state of the environment (e.g., "the block is on the table"). An action can only be performed if all of its preconditions are true statements about the current state of the environment. When an action is performed, the environment is modified (as necessary) to ensure that all of the action's effects will be true statements about the environment after the modifications are complete. A planning problem is specified by a providing a set of actions, the current state of the environment, and a desired goal state for the environment to be in. A planner is used to find a partial order over nodes such that executing each node's action once (while respecting the partial order) will cause the environment to adopt the desired goal state by the time the execution ends. A more detailed description of planning can be found in standard AI texts (Russel and Norvig 2010).

A partial-order plan can be thought of as a collection of possible stories. Nodes represent the ways in which the story can proceed from one state to the next (e.g., via non-player characters performing actions in the environment), and edges represent constraints on the order in which actions can be performed. By executing the actions of nodes in different orders (among those that are allowed by the edges), different state/action sequences can be obtained, each of which can be thought of as a different story in the environment. As a formulation of experience management, creating a partial-order plan is underspecified, because such plans do not distinguish between the actions that a player might perform and the ways in which the manager can influence the player's experience while it is underway. If every node in the plan is thought of as a potential player action, then the manager's influence on the experience is lost, aside from the initial creation of the partial order. Alternatively, if every node is thought of as a potential manager operation, then the environment would become non-interactive (unless other actions were made available). To ensure that the environment remains interactive while still allowing the manager to influence player experiences at runtime, the planning formulation must be extended. I discuss two such extensions in the following subsections: narrative mediation, and plan selection.

Narrative Mediation. The group of managers created by Young et al. (2004), Harris & Young (2005), and Riedl & Stern (2006) all extend the planning formulation in the same way: they assume that all of the actions in the plan will be carried out automatically by the manager (perhaps by controlling story characters), but that the player may independently perform other actions in the environment that might falsify the preconditions of actions in the plan. For example, the manager might need to execute an action (as part of the partial order plan) that causes an in-game character to "lift the block", but the player might destroy the block before this action can occur. When such a problem occurs, the managers in this group change the current plan that is being pursued to account for the player's action, either by solving a new planning problem from that moment forward (Young et al. 2004; Harris and Young 2005), or by switching to a new plan that was computed before the player's experience began (Riedl and Stern 2006). This type of manager behaviour is called *narrative mediation*. The manager by Harris & Young additionally attempts to detect such problems before they can influence the world state (e.g., after the player attempts to destroy the block, but before it gets destroyed) and then temporarily modify the player's action to ensure that its effect does not occur (e.g., the player's "destroy the block" action would fail). Although this method of extending the planning formulation of experience management distinguishes between the player's actions and the manager's influence over the experience, it does so by defining two kinds of manager operations: those that occur as part of the partial-order plan, and those which switch to a different plan in response to problematic player actions. Meanwhile, the formulation that I presented in Chapter 2 captures everything that a manager does in a single operation (changing the transition function of the player's environment), offering a more concise description. Furthermore, since the player's actions only ever influence the partial-order plan when they disrupt it by falsifying its preconditions, some players might have no effect on the story at all (which defeats the purpose of an interactive story). In Chapter 4, I describe how GEM managers can be created to ensure that every player's actions have some effect (via the manager) over the course of their experience in a given interactive environment.

Plan Selection. The managers created by Ramirez & Bulitko (2014) and Poo Hernandez et al. (2014) extend the planning formulation of experience management in a different way: they assume that all of the actions in the plan will be carried out by the player, except when only one action can be performed (due to the constraints imposed by the partial order plan); in this case, the action occurs automatically. In addition to the actions in the plan, however, players are also allowed to perform other actions in the environment that might falsify the preconditions of actions in the plan. When this happens, an extension of Riedl & Stern's (2006) manager is used to dynamically find a set of partial order plans that all account for the player's action. From this set, the plan whose actions best match either a learned model of the player's playing style (Ramirez and Bulitko 2014) or a designer-specified curve of the player's emotions over time (Poo Hernandez et al. 2014) is chosen by the manager and the experience proceeds using that plan. To ensure that every player's experience will be affected by the manager while it is underway, both managers can occasionally force certain player actions to invalidate the current story plan. That said, the managers that can be built in my proposed framework are more flexible still: they can choose a new transition function after every player action, without needing to monitor a set of plan step preconditions.

3.2 Dynamic Difficulty Adjustment

Dynamic Difficulty Adjustment (DDA) is a subfield of experience management that aims to optimize player experiences in environments where players pursue a given goal (e.g., winning a race, rescuing a character, or escaping from a group of enemies). The primary concern of such managers is to prevent players from feeling either of two emotions: frustration from finding their goal too difficult to achieve, or boredom from finding it to easy to achieve. By dynamically assessing the amount of difficulty that the current player is experiencing and then modifying the game to influence this value (e.g., by varying the speed of the player's opponents in a race), DDA managers work to ensure that the game's difficulty is always well suited to the current player's level of skill.

While dynamic difficulty adjustment has been used by commercial video games in an *ad hoc* manner for at least two decades[1], Hunicke & Chapman (2004) were among the first to study DDA in a more general sense. In their approach, the player is an agent acting in an interactive environment, and an external manager repeatedly observes the state of the environment while the game is underway. The manager uses an evaluation function to estimate the player's performance (i.e., their competence at pursuing in-game goals) and compares this value to a designer-specified curve of performance over time. The manager's policy then maps these values to a set of ways in which the environment's dynamics should be changed, toward causing the player's performance to follow the specified curve. As a formulation for experience management, Hunicke & Chapman's approach is similar to the formulation that I presented in Chapter 2 in that the manager changes the dynamics

[1]The game *X-COM: UFO Defense* (Mythos Games and MicroProse 1994) offers one example, where the difficulty of overcoming an invasion is automatically varied based on the player's performance at doing so.

(i.e., the transition function) of the player's environment. Unlike my formulation, however, their manager only observes the current world state to make its decisions. This makes Hunicke & Chapman's approach unsuitable to serve as a general framework for experience management, since some managers need to additionally consider its own previous decisions, the player's previous actions, or the environment's previous states. In addition to its ability to represent managers that consider entire histories of the player's experience (recall Section 2.1.1), the framework that I present in Chapter 4 contains components that support the evaluation function and designer-specified performance curve that Hunicke & Chapman describe (an objective function and a feature vector, respectively).

More recently, Zook & Riedl (2012; 2014) have extended Hunicke & Chapman's work by allowing the manager to automatically construct a temporal model of the player's performance – that is, a model that aims to capture how a player's performance changes over time as they learn better ways to achieve their given goals. The manager then uses this model to predict the player's future performance, toward changing the environment pre-emptively to avoid them becoming bored or frustrated. When considered as a formulation for experience management, Zook & Riedl's approach is more suitable than Hunicke & Chapman's because it allows the manager's decisions to depend on observations of the player's previous actions and the states of the environment in which they were performed. However, the manager cannot base its decisions on its own prior behaviour (e.g., toward ensuring that it influences the environment in a varied way). The framework that I present in Chapter 4 supports the creation of managers that can examine their prior decisions (as I described in the previous paragraph), and it also contains components for building models of the player and predicting future player actions (a feature vector and an estimated player policy, respectively).

3.3 Existing Frameworks

Roberts (2011) has proposed a framework for "fully-realized experience management" in which he identifies seven considerations for an AI manager's design. Four of the seven concern representing: (i) the content of an interactive environment, (ii) the actions that the manager uses to change each player's experience, (iii) information about how players are likely to act in the given environment, and (iv) the intended effects of the experience. The remaining three concern the computation that the manager will perform, namely: (v) selecting short-term subgoals to pursue that make progress toward achieving the intended effects of the experience, (vi) determining which of its available actions is the best for realizing its short-term goal, and (vii) ensuring that its actions will be narratively consistent with respect to the current state of the environment. While all of these considerations are important when designing an AI manager, Roberts has thus far only used them to describe managers in the context of Drama Management, and has not shown whether or not they can also be applied in the context of Dynamic Difficulty Adjustment. Furthermore, it remains unclear how the results of addressing these concerns should then be combined to create a working manager. In Chapter 4, I describe how all of GEM's components can be used together to manage a player's experience. In

Chapter 5, I demonstrate how GEM can be used to represent managers from the subfields of both Drama Management and Dynamic Difficulty Adjustment.

Summary

In this chapter, I discussed a variety of different formulations of experience management, each of which have been used to develop one of two kinds of managers: those that aim to optimize the player's experience in an interactive story, and those that aim to optimize the amount of difficulty that the player experiences while they play. For each different formulation, I described how its lack of either generality (e.g., being tightly integrated to a domain) or specificity (e.g., not describing how the manager can affect the player's experience) makes it unsuitable to use as a general framework for experience management. At the same time, I noted how the different components of the framework that I propose in Chapter 4 can support the different abilities that the managers that I discussed could perform, such as simulating potential futures, predicting the player's behaviour, or building a model of the player based on features of their experience thus far.

Chapter 4

Generalized Experience Management

In this chapter, I present Generalized Experience Management (GEM), the framework that I created to simplify understanding the diverse approaches to experience management that have previously been explored. I begin by describing GEM as an extension of the formulation of experience management that I presented in Chapter 2, and follow with a detailed example. I demonstrate GEM's versatility in Chapter 5 by using it to represent four different managers.

4.1 Foundation & Building Blocks

As I explained in Chapter 1, a manager is an agent that tries to optimize a player's experience in an interactive environment by modifying the dynamics of that environment. More formally, when an interactive environment is represented as a Markov Decision Process (MDP; $e = \langle S, A, (\tau_0, \dots) \rangle$) in which the player is an agent, experience management is the process of changing the MDP's transition function after each action that the player performs, toward maximizing that player's return (ξ) over the $n + 1$ time steps of their experience:

$$\tau_{t+1} = \operatorname*{arg\,max}_{\tau \in \mathcal{T}} \xi(h_n, \boldsymbol{r_n}) \tag{4.1}$$

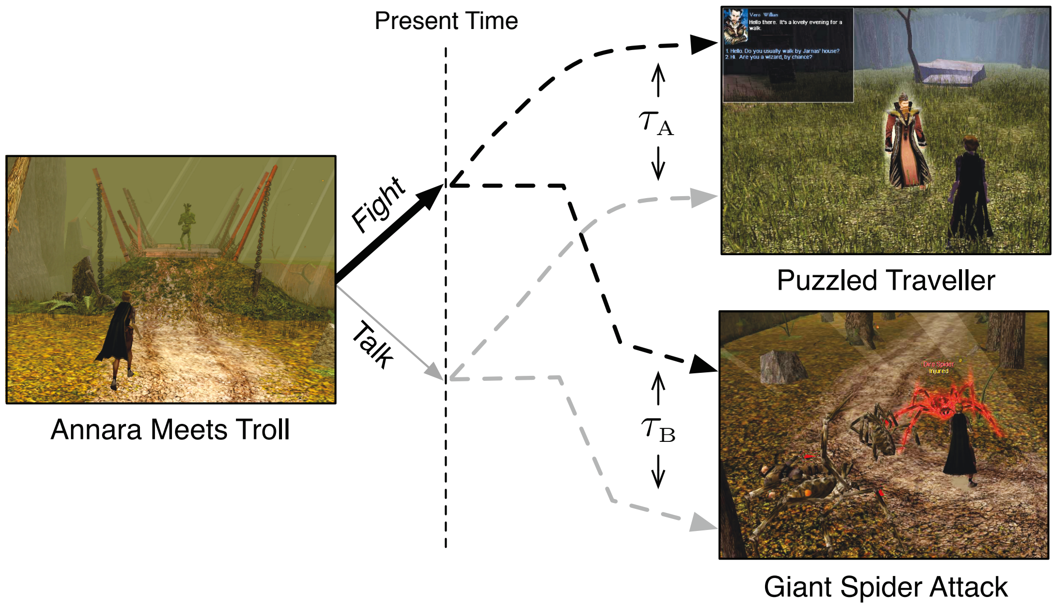
As I explained in Chapter 2, this equation is not computable because neither the return function (ξ) nor the vector of rewards that the player receives ($\boldsymbol{r_n}$) are ever known or controllable by the manager, and the complete history of the player's experience (h_n) is also unknown at the time that the computation is needed (time step $t + 1$, where $t < n$).

$$\tau_{t+1} = \chi(e_t, h_t) \tag{4.2}$$

While Equation 4.1 represents the fundamental *goal* of experience management, Equation 4.2 shows how little a manager might receive to perform this task: it could be required to choose a transition function for the current time step (using its policy, χ) based on only the current interactive environment ($e_t = \langle S, A, \tau_t \rangle$) and a history of the player's experience (h_t). Equation 4.2 represents the core of GEM, serving as a minimal representation for an experience manager's policy. Moving beyond this minimum, GEM is a framework in the sense that it organizes the common concepts of experience management into a well-defined, formal structure, providing both a foundation (Equation 4.1) and a set of conceptual "building blocks" (Sections 4.1.1 to 4.1.5) with which solutions to the problem of experience management can be made. I describe five kinds of building blocks in the following sections. Although all of them have been used in various forms in prior research, their integration into a cohesive, mathematical framework for experience management (the GEM framework) is a novel contribution of this work.

4.1.1 Block #1: Decision Constraints

Given an interactive environment $e = \langle S, A, (\tau_0, \ldots)\rangle$ in which player experiences will be managed, the designer might want to assert that only a reduced subset of the transition functions in $\mathcal{T}$ should be available for selection following particular player histories. For example, in the scene from *Annara's Tale* that I introduced in Section 1.1 (now shown in Figure 4.1), the designer has asserted that, of all the transition functions that *could* be used by the game after Annara begins to fight the Troll, only two of them should be available for the manager to choose from: τ_{A} and τ_{B}. The designer has also asserted that the same set should be available after Annara talks to the Troll instead.



Vero Willian
Hello there. It's a lovely evening for a walk.
1. Hello. Do you usually walk by Jarnas' house?
2. Hi. Are you a wizard, by chance?

Figure 4.1: An opportunity to change the player's experience in *Annara's Tale*, revisited. The player chooses to fight the troll that blocks Annara's path (thick straight arrow), and the manager can choose whether the game's transition function should be changed to either τ_{A} (curved dashed arrows) or τ_{B} (angular dashed arrows).

I refer to such restrictions as *decision constraints* because they constrain the manager's decisions. They can be specified using a *decision constraint function* (κ) that maps from player histories to non-empty subsets of $\mathcal{T}$ (i.e., $\kappa : H \to 2^{\mathcal{T}} \setminus \{\emptyset\}$). Table 4.1 shows a partial decision constraint function in the context of *Annara's Tale*. Of all transition functions that *could* be used in this environment (i.e., in $\mathcal{T}$), the manager can only choose between both τ_{A} and τ_{B} when the player chooses to either fight or talk to the troll.

History (h_t)	Decision Constraints ($\kappa(h_t)$)
($\ldots$, Annara Meets Troll, *Fight*)	$\{\tau_{\text{A}}, \tau_{\text{B}}\}$
($\ldots$, Annara Meets Troll, *Talk*)	$\{\tau_{\text{A}}, \tau_{\text{B}}\}$

Table 4.1: A tabular representation of a decision constraint function for two potential histories.

The ellipses ($\ldots$) in Table 4.1 demonstrate a form of abstraction that can simplify specifying the decision constraint function. Specifically, they are used in the table to denote that *every* history ending with the state/action pair $\langle$Annara Meets Troll, *Fight*$\rangle$ should be mapped to the given set of transition functions (and similarly for $\langle$Annara Meets Troll, *Talk*$\rangle$).

$$\tau_{t+1} = \operatorname*{arg\,max}_{\tau \in \kappa(h_t)} \xi(h_n, \boldsymbol{r_n}) \tag{4.3}$$

Equation 4.3 shows how the goal of experience management (Equation 4.1) can be reframed using a decision constraint function, where the set considered by the $\arg\max$ operator has been changed from $\mathcal{T}$ to $\kappa(h_t)$.

$$\tau_{t+1} = \chi(e_t, h_t, \kappa) : \tau_{t+1} \in \kappa(h_t) \tag{4.4}$$

Equation 4.4 shows how κ can be used to extend GEM's foundation (Equation 4.2) with the constraint that the selected transition function τ_{t+1} must be an element of the set given by $\kappa(h_t)$.

Decision Points

For convenience when discussing managers in later sections, I define a *decision point* as any state/action pair $\langle s_t, a_t \rangle$ after which more than one transition function is available for selection following some history $h_t \in H$ (i.e., where $|\kappa(h_t)| > 1$). Let $D \subseteq S \times A$ be the set of all decision points that exist for manager m in a given environment e. For example, two decision points are shown in Figure 4.1: $\langle$Annara Meets Troll, *Fight*$\rangle$ and $\langle$Annara Meets Troll, *Talk*$\rangle$. When no explicit decision constraints are used, every state/action pair $\langle s, a \rangle \in S \times A$ is a decision point, because the manager selects from $\mathcal{T}$ at each time step and $|\mathcal{T}| > 1$.

Discussion

Decision constraints can simplify the task of experience management by reducing the number of transition functions that the manager needs to consider when trying to maximize the player's return. Furthermore, using decision constraints sacrifices no generality: when $\kappa(h_t) = \mathcal{T}$ for every $h_t \in H$, Equation 4.3 is equivalent to Equation 4.1 (the goal of experience management), and Equation 4.4 is equivalent to Equation 4.2 (the foundation for GEM).

When specifying decision constraints in practice, the potentially infinite size of H can be handled by asserting that unless otherwise specified, the manager will keep the transition function constant (i.e., $\kappa(h_t) = \{\tau_t\}$, where $h_t = (\ldots, \tau_t, s_t, a_t)$).

Decision constraints also allow the designer to exercise complete control over each player's experience by reducing the output of $\kappa(h_t)$ to a single-element set for any history h_t.The only aspect of a manager's operation that cannot be controlled through decision constraints is how the manager selects transition functions when the set given by $\kappa(h_t)$ contains more than one element; this process depends on the manager's policy. For example, while decision constraints can be used to force a manager to choose between three particular transition functions (e.g., $\{\tau_\text{A}, \tau_\text{B}, \tau_\text{C}\}$), they cannot be used to control *how* the manager makes its selection (e.g., by a weighted random draw, an expansion of possible futures using each available function, or any other method). In general, decision constraints provide a convenient way for designers to distinguish between transition functions that should and should not be considered by the manager's policy following certain player histories. Being able to make this distinction is useful because it allows the designer to eliminate any transition functions that would *always* be detrimental to maximizing the player's return (e.g., a transition function that maps every state/action pair of *Pac-Man* (Namco 1980) to one in which the player has zero remaining lives). To move beyond this binary evaluation of each transition function, however, an objective function is required.

4.1.2 Block #2: An Objective Function

Recall from Chapter 2 that a manager chooses transition functions in an attempt to dynamically maximize each player's return. Given that the player's reward function and return are both inaccessible, some managers *estimate* the player's return using a designer-provided *objective function*. Formally, a manager's objective function (ϕ) maps from a transition function ($\tau \in \mathcal{T}$), a history ($h_t \in H$), and the current interactive environment ($e_t \in E$) to a real number ($\phi : \mathcal{T} \times H \times E \to \mathbb{R}$). Equation 4.5 shows how an objective function (ϕ) can be used to pursue the goal of experience management by defining the manger's policy (Equation 4.2) as an approximation of Equation 4.1, where the player's return (ξ) has been replaced by ϕ:

$$\tau_t = \chi(e_t, h_t, \phi) = \operatorname*{arg\,max}_{\tau \in \mathcal{T}} \phi(\tau, h_t, e_t) \tag{4.5}$$

For example, suppose that the intended effect of *Annara's Tale* is for players to have fun. To pursue this effect using an experience manager in the scene from Figure 4.1, the designer could create an objective function (ϕ) that estimates how much fun the player would have if e_t's transition function were to switch to each available τ following each history h_t. Table 4.2 shows an example of how part of this objective function might be specified. Reading across the top row of the table, this objective function states that whenever the player's history ends with the player meeting and choosing to fight the Troll, selecting transition function τ_A is expected to result in an experience (i.e., an alternating sequence of states and actions) that yields 20 arbitrary "points" of fun, while selecting τ_{B} is expected to yield 30 points.

History (h_t)	Transition Functions ($\tau \in \mathcal{T}$)	Estimated Fun (ϕ)
(…, Annara Meets Troll, *Fight*)	τ_{A}	20
	τ_{B}	30
(…, Annara Meets Troll, *Talk*)	τ_{A}	40
	τ_{B}	10

Table 4.2: A tabular representation of an objective function for two potential histories. For a given history and transition function, this function estimates how much fun the current player would have if the game's transition function were to change to the given one after the given history occurred.

When the player chooses to fight the troll in the example from Figure 4.1 (triggering decision point $\langle$Annara Meets Troll, *Fight*$\rangle$ in the table), the manager examines each of the available transition functions using its objective function, and chooses the one with the highest estimated value ($\chi(e_t, h_t, \phi) = \tau_{\text{B}}$, according to the partial information shown in Table 4.2). The game's transition function is then set to the manager's choice ($\tau_{t+1} = \tau_{\text{B}}$), and this new transition function is used by the environment to bring the game into its subsequent state.

Discussion

Maximizing an objective function (e.g., Equation 4.5) sacrifices no generality with respect to how τ_{t+1} is selected at each time step. To understand why, consider a function that is capable of representing any method of selecting a single transition function τ_{t+1} following history h_t: namely, let ζ be a function that maps from a set of transition functions, the player's history, and the current environment to a single transition function ($\zeta : 2^{\mathcal{T}} \times H \times E \to \mathcal{T}$). By defining an objective function $\phi(\tau, h_t, e_t)$ as an indicator function over $\mathcal{T}$ that gives 1 when $\tau = \zeta(\mathcal{T}, h_t, e_t)$ and 0 otherwise (Equation 4.6), one can ensure that the transition function that ζ selects will always be the one that maximizes ϕ in Equation 4.5, and thus gets selected by the manager's policy (χ).

$$\phi(\tau, h_t, e_t) = 1\big(\tau = \zeta(\mathcal{T}, h_t, e_t)\big) \tag{4.6}$$

For example, this strategy could be used to select transition functions using the $\operatorname{softmax}$ operator shown in Equation 4.7, which gives a transition function by sampling randomly from the probability distribution over $\mathcal{T}$ that is shown in Equation 4.8 (Sutton and Barto 1998) (λ is an arbitrary, real-valued function).

$$\zeta(\mathcal{T}, h_t, e_t) = \operatorname*{softmax}_{\tau \in \mathcal{T}} \lambda(\tau, h_t, e_t) \tag{4.7}$$

$$\Pr(\tau|\lambda) = \frac{e^{\lambda(\tau, h_t, e_t)}}{\sum_{\tau' \in \mathcal{T}} \left(e^{\lambda(\tau', h_t, e_t)}\right)} \tag{4.8}$$

Intuitively, the higher the value given by $\lambda(\tau, h_t, e_t)$, the more likely it is that τ will be selected by the $\operatorname{softmax}$ operator, and then subsequently be (uniquely) mapped to the value 1 when computing the objective function shown in Equation 4.6. The indicator function will map all other transition functions to the value 0, meaning that maximizing the objective function will result in the correct transition function being selected by the manager's policy (i.e., the one that was selected by the $\mathrm{softmax}$ operator in Equation 4.7).

The objective function's estimate of the player's return for each transition function need not be represented in an *absolute* sense; the policy's $\arg\max$ operator would be equally effective if the objective function provided a reliable *ranking* over the available transition functions.

Although using an objective function gives a designer complete control[1] over how each transition function will be selected following any player history, it is still *conceptually* convenient to use decision constraints (Block #1) even when an objective function is also used. Specifically, using both blocks allows the designer to clearly distinguish between the (reduced) *domain* of each manager decision (as given by decision constraints) and the *mapping* from this domain to an estimate of the player's return (as given by the objective function). This distinction allows the designer to focus their effort on distinguishing carefully only between the transition functions that rise above a certain (intuitive) threshold with respect to maximizing the player's return: decision constraints eliminate every transition function that falls below the threshold, and the objective function ranks the rest according to how will they will maximize the player's return.

4.1.3 Block #3: A Rollout Function

As an estimate of the player's return function (ξ in Equation 4.1), the objective function as stated in Equation 4.5 can be improved: while the player's return is based on the player's entire history (i.e., h_n, assuming an $n + 1$ time step experience) the objective function in Equation 4.5 only receives information about the player's history prior to the current time step t (i.e., h_t) along with a candidate transition function to use as τ_{t+1}. To improve the objective function's estimates, some managers use a *rollout function* to create a set of trajectories that extend from the current decision point ($\langle s_t, a_t \rangle$) forward into the future of the player's experience, assuming that the transition function that is given to the objective function will be used as τ_{t+1}.

Formally, a rollout function (ρ; "rho" for "rollout") maps from a decision point ($\langle s, a \rangle \in S \times A$), a transition function ($\tau \in \mathcal{T}$), and an interactive environment ($e_t \in E$) to a set of trajectories that could potentially follow after the given point occurs and τ is selected as τ_{t+1} (i.e., $\rho : S \times A \times \mathcal{T} \times E \to 2^H$). I refer to each member of the set given by the rollout function as a *rollout*: $h^+ \in \rho(s_t, a_t, \tau, e_t)$.

For example, consider the decision point in *Annara's Tale* where Annara is alone in the forest and decides to explore ($\langle$Alone in Forest, *Explore*$\rangle$; see Figure 4.2)[2]. Suppose that the manager is using an exhaustive rollout function – that is, one that generates every possible trajectory that could occur (for some player) following a given decision point. Looking into the future from the

[1]Modulo the random tie-breaking of the $\arg\max$ operator.

[2]In this example, the transition functions τ_{A} and τ_{B} are deterministic (i.e., for each state/action pair $\langle s, a \rangle$, $\tau_{\mathrm{A}}(s, a, s') = 1$ for exactly one state s', and 0 for all other states).

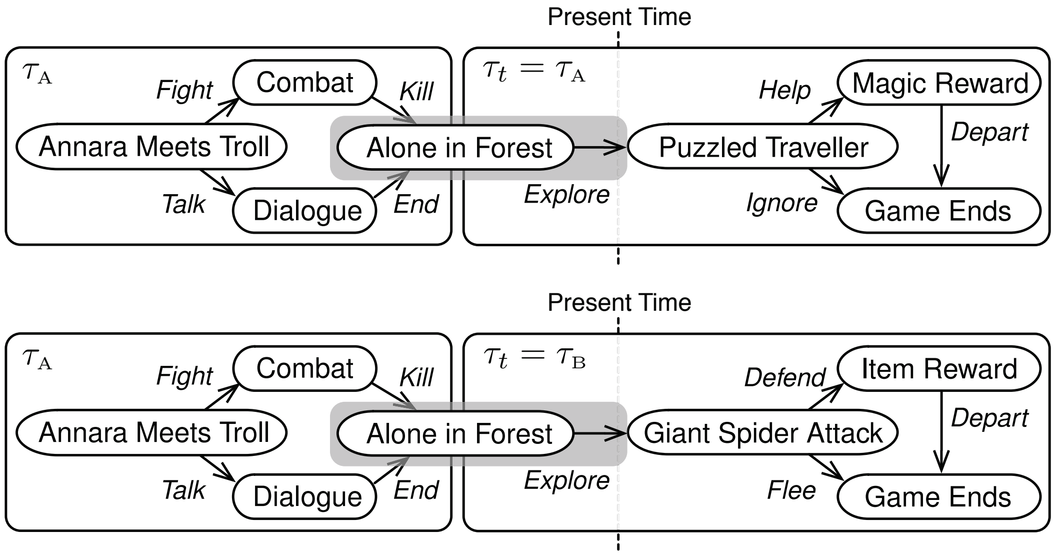


Figure 4.2: Part of *Annara's Tale* represented as an MDP, showing the effects of choosing one of two available transition functions (τ_A or τ_B) at the highlighted decision point. Ovals are game states, arrows show the combined effect of a single player action and a deterministic transition function. Top: The MDP if τ_A were selected. Bottom: The MDP if τ_B were selected.

current decision point, the (exhaustive) set of possible futures that could occur depends on whether the manager selects τ_A or τ_B as τ_{t+1}. Table 4.3 shows the result of the rollout function in both cases: two trajectories could occur following τ_A, and two different trajectories could occur following τ_B.

τ	Rollouts ($h^+ \in \rho(s_t, a_t, \tau, e_t)$)
τ_A	(Puzzled Traveller, *Help*, τ_A, Magic Reward, *Depart*, τ_A, Game Ends)
	(Puzzled Traveller, *Ignore*, τ_A, Game Ends)
τ_B	(Giant Spider Attack, *Defend*, τ_B, Item Reward, *Depart*, τ_B, Game Ends)
	(Giant Spider Attack, *Flee*, τ_B, Game Ends)

Table 4.3: Possible rollouts for each of two given transition functions, starting from $\langle s_t, a_t \rangle = \langle$Alone in Forest, *Explore*$\rangle$ in Figure 4.2.

Equation 4.9 demonstrates a simple way to use a rollout function to compute the result of an objective function, where $\lambda : H \to \mathbb{R}$ denotes an arbitrary function related to the player's return:

$$\phi(\tau, h_t, e_t, \rho) = \max_{h^+ \in \rho(s_t, a_t, \tau, e_t)} \lambda((h_t, h^+)) \tag{4.9}$$

This objective function identifies the rollout which, when appended to the player's history, creates the longer, hypothetical trajectory ((h_t, h^+)) that maximizes the given function (λ); this maximum value is then returned as the value of the objective function.

Discussion

While exhaustive rollout functions return a set containing every possible future (like the one in the example above), such functions may not be tractable to compute. In general, any algorithm that identifies different trajectories in an MDP can be used, including those which only expand possible futures up to a finite horizon (which could make them more practical).

When computing any rollout function, transition functions must be determined along with states and actions. The rollout function must determine how these choices should be made, which might mean using the manager's policy once for each decision point that is reached during the expansion of possible futures.

4.1.4 Block #4: An Estimated Player Policy

In addition to retrieving a set of possible futures that could occur following a particular state/action pair, it can be useful for the manager to predict the probability with which each possible future will occur (toward computing the expected value of each future; see Section 4.2). While the transition function at each time step allows the prediction of subsequent states, predicting subsequent actions requires an additional function. An *estimated player policy* ($\tilde{\pi}$) is a function that maps from a given history $h_t \in H$, a state $s \in S$, and an action $a \in A$ to an estimate of the probability that the player will perform action a in state s given that h_t has occurred earlier in the game ($\tilde{\pi} : H \times S \times A \rightarrow [0, 1]$). Including the player's history as a parameter allows the manager to vary its estimate of the player's policy based on their previous actions in the game (i.e., the player's policy can be learned).

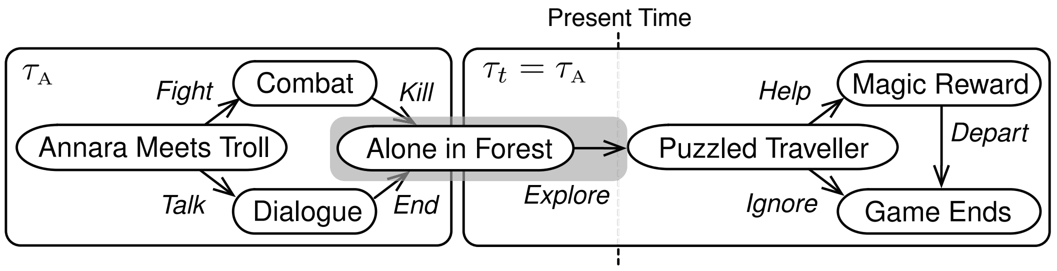


Figure 4.3: Part of the MDP for *Annara's Tale*. Ovals are game states, arrows show the combined effect of a single player action and a deterministic transition function. The symbol τ_{A} identifies the game's current transition function.

Table 4.4 shows part of an estimated player policy for the section of *Annara's Tale* that appears in Figure 4.3. If the given history includes the player having chosen to fight the troll, then this policy predicts a 10% chance that the player will help the puzzled traveller, and a 90% chance that they will ignore him instead. On the other hand, if the history includes the player having chosen to talk to the troll instead, then the policy predicts an 80% chance that the player will choose to help the traveller, and a 20% chance that they will ignore him.

History	State	Action	Probability
$(\ldots$, Annara Meets Troll, *Fight*, τ_{A}, Combat, *Kill*, τ_{A}, Alone in Forest, *Explore*$)$	Puzzled Traveller	*Help*	0.1
		Ignore	0.9
$(\ldots$, Annara Meets Troll, *Talk*, τ_{A}, Dialogue, *End*, τ_{A}, Alone in Forest, *Explore*$)$	Puzzled Traveller	*Help*	0.8
		Ignore	0.2

Table 4.4: Part of a tabular representation of an estimated player policy. For a given history, state, and action, the estimated player policy gives a probability that the player will perform the action in that state, given that history h_t has occurred earlier in the game.

This example demonstrates how $\tilde{\pi}$'s estimates of the player's actions can depend on the player's history. Although this example shows only estimates for how the player will act in the immediately subsequent state (i.e., s_{t+1}), an estimated player policy can generally be used to estimate the player's action in any given state.

$$\chi(e_t, h_t, \phi, \tilde{\pi}) = \arg\max_{\tau \in \mathcal{T}} \phi(\tau, h_t, e_t, \tilde{\pi}) \tag{4.10}$$

Equation 4.10 shows how an estimated player policy can be used as part of a manager policy, where the estimated player policy ($\tilde{\pi}$) is given as an extra input to the objective function.

Discussion

In general, the probability of any given rollout occurring following a particular player history is given by Equation 4.11. The second and third lines expand the right side of the first line using Bayes' Rule, and the fourth line simplifies each term (e.g., the occurrence of a_{t+1} does not depend on τ_{t+1} or χ, so these terms can be dropped from the probability of a_{t+1}).

$$\begin{aligned}
\Pr(h_{t+1}|h_t, \chi, e_t, \tilde{\pi}) &= \Pr(\tau_{t+1}, s_{t+1}, a_{t+1}|h_t, \chi, e_t, \tilde{\pi}) \\
&= \Pr(s_{t+1}, a_{t+1}|\tau_{t+1}, h_t, \chi, e_t, \tilde{\pi}) \Pr(\tau_{t+1}|h_t, \chi, e_t, \tilde{\pi}) \\
&= \Pr(a_{t+1}|s_{t+1}, \tau_{t+1}, h_t, \chi, e_t, \tilde{\pi}) \Pr(s_{t+1}|\tau_{t+1}, h_t, \chi, e_t, \tilde{\pi}) \\
&\quad \times \Pr(\tau_{t+1}|h_t, \chi, e_t, \tilde{\pi}) \\
&= \Pr(\tau_{t+1}|h_t, \chi, e_t) \Pr(s_{t+1}|\tau_{t+1}, h_t, e_t) \Pr(a_{t+1}|s_{t+1}, h_t, e_t, \tilde{\pi}) \\
&= \Pr(\tau_{t+1}|\chi, h_t, e_t)\tau_{t+1}(s_t, a_t, s_{t+1})\tilde{\pi}(h_t, s_{t+1}, a_{t+1})
\end{aligned} \tag{4.11}$$

In the last line of this equation, $\Pr(\tau_{t+1}|\chi, h_t, e_t)$ denotes the probability of the manager's policy (χ) selecting the transition function that is being considered for time step $t+1$ (i.e., τ_{t+1}), given history h_t and environment e_t.

4.1.5 Block #5: A Feature Vector

Instead of having an objective function or a decision constraint function depend directly on the player's history, it can be more convenient for the designer to define one or more *features* of the player's history for these functions to use.

As I described in Section 4.1.2, an objective function estimates the player's return (ξ), which represents how successfully that player was affected by their experience in the way that the designer intended (recall Section 2.1.2). For any given intended effect, there might be several factors of the experience that influence this success (e.g., the believability of non-player characters, or the pacing of exciting situations). Whenever the designer can identify factors of the player's experience that affect how successful it might be, the task of specifying the objective function can be simplified by explicitly representing each factor as a *feature*. A feature is a function that maps from a given trajectory to a real number ($f : H \to \mathbb{R}$), and a group of n such features can be represented conveniently by a length n *feature vector* ($\boldsymbol{f} : H \to \mathbb{R}^n$). By changing the total number of features or the range of each feature, the designer can set the amount of abstraction with which each feature vector represents its associated state/action history.

History	Feature Values	
	Fighter	Storyteller
(…, Annara Meets Troll, *Fight*, τ_A, Combat, *Kill*, τ_A, Alone in Forest, *Explore*)	65	0
(…, Annara Meets Troll, *Talk*, τ_A, Dialogue, *End*, τ_A, Alone in Forest, *Explore*)	0	40

Table 4.5: Part of a tabular representation of a simple feature vector. For a given player history, the feature vector gives a vector of real numbers that contains a value for each feature. Only two possible histories are shown.

Table 4.5 shows an example of a feature vector of length 2; one feature (f_Fighter) represents the designer's estimate of how inclined the player is toward performing fighting actions during the game, while the other feature ($f_\text{Storyteller}$) represents a similar estimate with regard to the player's inclination toward seeking more information about the game's story. From the (partial) history where the player has had Annara fight and kill the Troll, the designer estimates that the player's inclination toward fighting has a value of 65 points, while their inclination toward storytelling remains at zero; a different history (with talking instead of fighting) results in a different estimate of the player's inclinations, and thus different feature values.

$$\chi(e_t, h_t, \phi, \boldsymbol{f}) = \arg\max_{\tau \in \mathcal{T}} \phi(\tau, h_t, e_t, \boldsymbol{f}) \tag{4.12}$$

$$\phi(\tau, h_t, e_t, \boldsymbol{f}) = \lambda(\boldsymbol{f}(h_t)) \tag{4.13}$$

Equation 4.12 shows an example of how a feature vector can be used as part of a manager's policy, where the vector is passed along as an extra parameter to an objective function. Equation 4.13 shows how the objective function itself can then be defined as an arbitrary function (λ) of the output of the feature vector (which is a vector of n real numbers): $\lambda : \mathbb{R}^n \to \mathbb{R}$.

Feature vectors can also be used in conjunction with a decision constraint function (κ). For example, κ could be defined for *Annara's Tale* as follows: for every player history h_t ending in

$\langle$Alone in Forest, *Explore*$\rangle$, if $f_{\text{Fighter}}(h_t) < f_{\text{Storyteller}}(h_t)$, then $\kappa(h_t) = \{\tau_{\text{A}}\}$; otherwise, $\kappa(h_t) = \{\tau_{\text{B}}\}$. The effect of defining κ this way is that players who had demonstrated stronger inclinations toward learning about the game world ("Storytellers") would encounter the puzzled traveller, while players inclined more toward combat ("Fighters") would witness the spider attack instead (recall Figure 4.2 and the example from Table 4.5); note that no objective function (Block #2) is needed.

Discussion

Although the example features that I presented in the previous section each mapped a player's *history* to a real number, features can generally be used to map any *trajectory* to a real number. They can thus be combined with a rollout function (Block #3) and an objective function (Block #2) to simplify estimating the player's return from a combination of both the player's history and any predicted rollout. For example, Equation 4.14 shows an extension of Equation 4.9, where the arbitrary function λ operates on the output of the feature vector as in Equation 4.13, and the feature vector now accepts a concatenation of the player's history (h_t) and a rollout from the rollout function (h^+ from ρ).

$$\phi(\tau, h_t, e_t, \rho, \boldsymbol{f}) = \max_{h^+ \in \rho(s_t, a_t, \tau, e_t)} \lambda(\boldsymbol{f}((h_t, h^+))) \tag{4.14}$$

When applied directly to a player history, feature vectors can simplify specifying a decision constraint function, since any discrete or continuous variable (e.g., a counter or a binary flag) can be represented by a single feature, and such variables can be useful when determining which transition functions should be available for selection (e.g., reserving the hardest difficulty setting of a game (τ_{Hard}) until at least the tenth level of a video game ($f_{\text{Level}}(h_t) \geq 10$)).

4.2 Building a Manager

GEM's building blocks are complementary, and can be used in conjunction with one another. Figure 4.4 demonstrates how these blocks can be used to create a new manager: start from the foundation that I described in Chapter 2, choose to use at least one of Blocks #1 and #2, and then add more blocks as desired to either further refine the manager's estimates of the player's return (e.g., by examining rollouts in the objective function) or to simplify the task of creating the manager itself (e.g., by defining the objective function or decision constraints in terms of only the most important features of the player's experience).

Equations 4.15 and 4.16 demonstrate how a combination of all five blocks could be represented using GEM, where the probability term in Equation 4.16 is defined by Equation 4.11, and λ represents an arbitrary, designer-defined function that maps the values in a feature vector to a real number that estimates the player's return ($\lambda : \mathbb{R}^n \to \mathbb{R}$).

$$\chi(e_t, h_t, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}) = \operatorname*{arg\,max}_{\tau \in \kappa(h_t)} \phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{4.15}$$

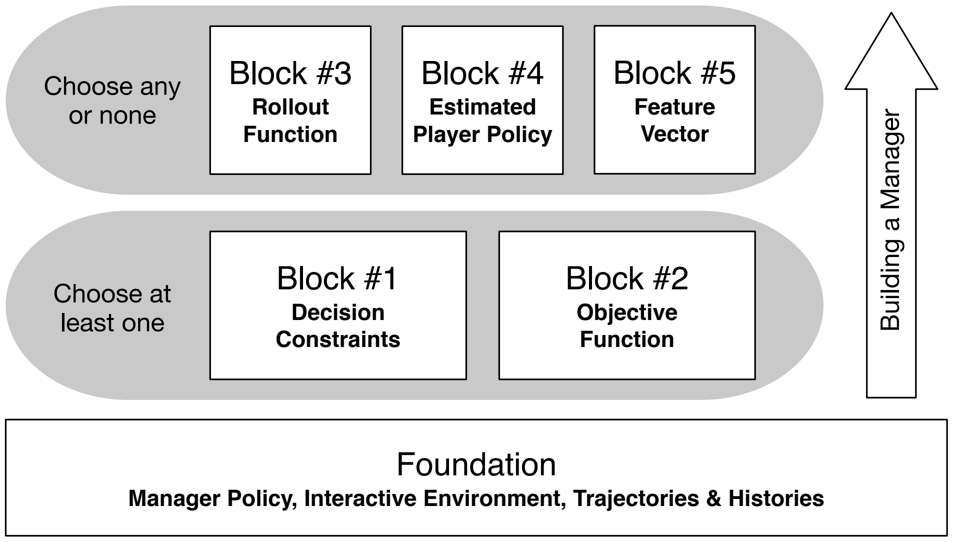


Figure 4.4: A diagram showing how GEM can be used to build a new manager.

$$\phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = \sum_{h^+ \in \rho(s_t, a_t, \tau, e_t)} \Pr(h^+ | h_t, \chi, \tilde{\pi}) \lambda\Big(\boldsymbol{f}\big((h_t, h^+)\big)\Big) \tag{4.16}$$

Taken together, these equations define a manager that chooses transition functions (from a set constrained by κ) to maximize its objective function (ϕ). The objective function's value is computed by sampling possible future trajectories (h^+) from the current decision point onward (via the rollout function, ρ) and calculating the product of two values for each rollout. One value is the probability of the rollout occurring (via an estimated player policy, $\tilde{\pi}$; recall Equation 4.11). The other value is an estimate of what the player's return would be if their combined experience ((h_t, h^+)) had certain designer-defined features ($\boldsymbol{f}$); this estimate is computed according to the designer-defined function λ. The result of this objective function is the *expected value* (in terms of λ) of selecting τ to continue the player's experience.

Summary

In this chapter, I presented Generalized Experience Management (GEM), a novel framework for representing experience managers. I explained GEM in terms of a foundation and building blocks, where each of five blocks serves a particular role in defining a manager's operation. *Decision constraints* (Block #1) allow a designer to restrict the set of transition functions that a manager can choose from following certain player histories. An *objective function* (Block #2) lets a designer specify how one transition function should be preferred when more than one is available for the manager to select, as an estimate of the player's return. A *rollout function* (Block #3) defines the mechanism that a manager uses to examine potential futures of the player's experience, toward improving the objective function's estimate. An *estimated player policy* (Block #4) allows the manager to predict the probabilities with which the futures that it generates might occur, toward further improving the objective function's estimate. Finally, a *feature vector* (Block #5) simplifies the specification of other blocks by allowing the designer to identify and compactly represent patterns in trajectories that are of particular importance when computing any of the other functions. I concluded by describing how the five blocks can be combined to create a new manager.

Chapter 5

Case Studies

In this chapter, I demonstrate the utility of Generalized Experience Management by using it to represent four different experience managers. The first two managers, Moe (Weyhrauch 1997) and the AI Director in *Left 4 Dead* (Valve Corporation 2008), were created by others independently from my work on GEM. For each of these managers, I first describe its operation using the terms of its authors, and then explain how it can be represented using GEM (for Moe, see Section 5.1; for the AI Director, see Section 5.2). I created the third manager, PaSSAGE (Thue 2007; Thue et al. 2007a; 2007b), during my prior work in this area, and I continued to study it during my development of GEM (Thue et al. 2008b; 2008a; 2009; 2010b). I created the fourth manager, PaSSAGE 2 (Thue et al. 2010a; 2011), during my development of GEM. For these two managers, I describe how each of its components can be represented using GEM immediately after I introduce that component (for PaSSAGE, see Section 5.3; for PaSSAGE 2, see Section 5.4).

5.1 Moe

Moe (Weyhrauch 1997) is one of the earliest examples of experience managers in the context of Drama Management. Designed to support the thesis that "interactive drama is possible", Weyhrauch created Moe to manage the experience of a single player in the context of the interactive drama *Tea for Three* (a murder mystery). Weyhrauch created *Tea for Three* based on a part of the interactive fiction *Deadline* (Blank 1982).

5.1.1 The Player's Experience

Moe represents each player's experience as a sequence that consists of two kinds of *moves*: *Moe moves* represent operations that the manager is able to perform, and *player moves*[1] represent particular player actions that are important to the progression of the experience. Examples of Moe moves include removing particular non-player characters (NPCs) from the game, causing NPCs to

[1]Weyhrauch used the phrase "user moves" to describe player actions, but I have changed their name to "player moves" to avoid any confusion that could arise from using multiple terms to discuss the same concept. Weyhrauch's "user" and the "player" that I describe in this work are the same entity.

travel from one game location to another, and changing what story information is revealed by certain player actions. Examples of player moves include discovering evidence, learning important story information, and confronting characters about the mystery (the player's role in the story is that of a police detective). The experience is generated by Moe and the user taking turns, and during their respective turns, both Moe and the user are free to perform any number of the moves that are legal at the point at which they are considered. For example, five player moves might be followed by three Moe moves, followed by one player move, followed by two Moe moves, and so on.

Determinations of which moves are legal at a given point in the experience are made differently for Moe moves and player moves. For each Moe move, a function called *LegalWhen* accepts the sequence of Moe and player moves that have occurred thus far and gives whether or not that move is legal. The set of legal player moves is determined by referring to a *plot graph*: a directed, acyclic graph whose nodes are player actions, and whose edges encode temporal precedence relationships that must be enforced between the connected actions. For example, according to the plot graph shown in Figure 5.1, the player move "Catch George" can only be performed after the player has confronted George in an earlier move, but the move "Mud" can be performed at any time. Moe can make further restrictions to the set of legal player moves by performing certain Moe moves (as I describe in the following section).

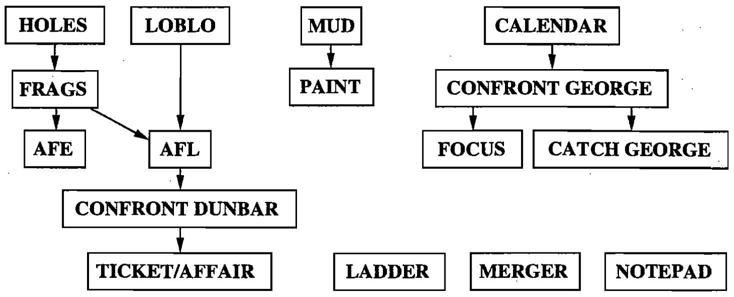


Figure 5.1: A plot graph of player moves from *Tea for Three*, the interactive drama managed by Moe. Source: (Weyhrauch 1997). Each player move at the tail of an arrow must be performed by the player before the player move at the head of that arrow will become legal to perform.

5.1.2 Moe's Operations

Whenever the player performs a player move during the experience, Moe is given an opportunity to select any number of legal Moe moves to occur. Weyhrauch grouped Moe's moves into six categories:

- *causers*, which force the player into performing a certain player move;
- *deniers*, which prevent the player from performing a certain player move;
- *hints*, which attempt to bias the player toward performing a certain player move;

- *move substitutions*, which force the player into performing one player move as a result of performing another (the player is also prevented from performing the former move directly);
- *delayers*, which (also) prevent the player from performing a certain player move[2]; and
- *future hints*, which attempt to bias the player toward performing a certain player move that will only become legal later on in the experience.

Each kind of move represents a relatively small change to the interactive environment.

5.1.3 Moe's Policy

The intended effect of *Tea for Three* is that each of its players should have a "dramatic, story-like experience" (Weyhrauch 1997; pg. 92); Moe's goal is to ensure that each player's experience in *Tea for Three* is maximally successful at affecting them in this intended way. Whenever Moe is required to choose a Moe move to occur, it receives as input the sequence of Moe moves and player moves that has occurred thus far, as well as a compact representation of the plot graph that determines which player moves are legal. Using this information, Moe simulates potential futures of the player's experience using a search that expands depth-first to every possible ending (an ending occurs when all player moves have been performed once each). More specifically, for each of the legal Moe moves that it has to choose from, Moe considers every complete future that could occur if it made that move. Moe uses a simple player policy to predict player actions (e.g., the user always follows hints) and it generates each future according to two assumptions: (i) that it will always be able to choose new Moe moves as needed (until the experience ends), and (ii) that its estimate of the player's policy should be updated as each new step of the (hypothetical) future is generated.

For each complete future, Moe estimates the amount of success that the player's experience as a whole would have, if it were to end with that future after having begun with the sequence of Moe moves and player moves that actually occurred in the game. Moe computes this estimate using an *evaluation function*, which calculates a weighted combination of a set of *features*: functions that examine the sequence of moves that has occurred thus far and each return a number between zero and ten. The value of each feature is computed from the given sequence using a collection of expert knowledge that is specific to that feature. For example, the *thought flow* feature examines the sequence of player moves that occurred in an experience, and considers which of five, predefined story topics each player move is related to (every player move is related to at least one topic). The value of thought flow for that experience is then computed as the number of consecutive pairs of player moves in the experience that share at least one related topic. In total, Weyhrauch defined seven features:

- *thought flow* estimates whether the given progression of player moves would seem logical;

[2]I suspect that Weyhrauch's distinction between deniers and delayers is due to his intention for every delayer to have a corresponding causer that re-enables the player move that it "delayed", while the player moves that deniers disable are never re-enabled. Functionally, the moves have identical effects.

- *activity flow* estimates the player's level of engagement with the story;
- *options* estimates how much freedom the player perceives;
- *motivation* estimates how well the player moves are motivated by the player's goals;
- *momentum* gives a bonus for certain player moves occurring soon after others;
- *intensity* estimates the player's excitement; and
- *manipulation* estimates the player's feeling of having been manipulated by the given sequence of Moe moves.

The first six of these features are computed from the sequence of player moves that appears in the given complete experience, while the manipulation feature is computed from the sequence of Moe moves. Once the value of each feature has been computed, the evaluation function calculates a weighted sum of the results using a set of designer-specified weights.

It is important to note that Moe's evaluation function only estimates how successful a *complete* experience might be (i.e., one in which every move has been used once). However, while the player's experience is underway, the sequence of Moe moves and player moves that will comprise their complete experience is not yet known. To estimate how much success will result from a *partial* experience (e.g., of the player's experience thus far plus one candidate Moe move), Moe examines possible futures of the experience to compute a backed-up estimate using a recursive "avg-max" approach, as follows. If adding one move to the current partial experience results in a complete experience, then its value is estimated using the evaluation function (no backup is required). Otherwise, the next possible move (following the current partial experience) will be either a Moe move or a User move. If it is a player move, then the value of the current partial experience is estimated as the expected value of the player performing a legal player move at that point in the experience; this is the "avg" (averaging) part of the approach. The weights for the expected value come from an estimated player policy. If the next possible move is a Moe move, then the value of the current partial experience value is estimated as the maximum of the estimated values of performing a legal Moe move following that partial experience; this is the "max" (maximizing) part of the approach.

5.1.4 GEM Representation

To demonstrate how Moe can be represented in terms of Generalized Experience Management, I consider each of Moe's components in turn.

Player Moves

In addition to describing the actions that players can perform (i.e., player actions in GEM ($a \in A$)), player moves define the points at which Moe gets activated to choose a Moe move. This activation occurs immediately following the performance of any player move, which is consistent with how each player action can be followed by a manager decision in GEM.

Moe Moves

Moe moves are very similar to the transition functions that are available in GEM's set of transition functions (i.e., $\tau \in \mathcal{T}$). However, while Weyhrauch's framework allows multiple Moe moves to occur in immediate succession, GEM always gives the player at least one opportunity to act after each transition function is selected and applied to the game. This difference can be reconciled with the help of two facts: (i) that each of the six categories of Moe moves can be represented as a change to a transition function (see Figures 5.2 to 5.4), and (ii) that changing a transition function multiple times still results in a single function. In each of the figures, circles show game states, solid arrows show player actions, and dashed arrows show how the current transition function (which is indicated in the upper left) maps each action from its originating state to a new state. The shaded areas show how the transition function can be changed from τ_1 to τ_2 to implement a Moe move. Thick edges show the states and actions that have occurred in the game thus far.

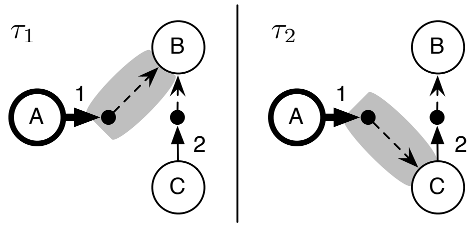


Figure 5.2: Representing a Moe *causer* or *hint* as changing a transition function. Changing the transition function from τ_1 to τ_2 could force the player to perform action 2 (the desired result of a *causer*). The same change could force the player to observe a *hint* in state C.

In Figure 5.2, a Moe *causer* that forces the player to perform action 2 can be implemented by changing the transition function from τ_1 to τ_2. Similarly, a Moe *hint* that forces the player to experience state C (which could deliver the desired hint) can be accomplished by the same change.

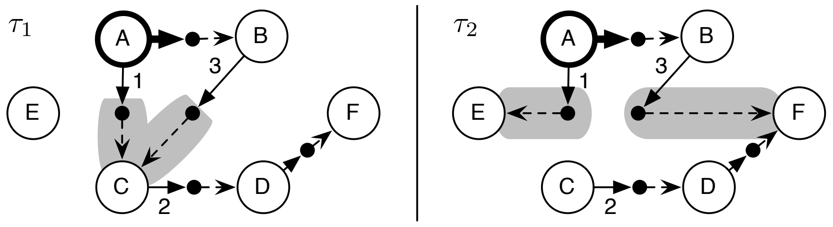


Figure 5.3: Representing a Moe *denier* or *delayer* as changing a transition function. When the transition function is changed from τ_1 to τ_2, none of the player's actions can cause the game to reach state C.

In Figure 5.3, a Moe *denier* or *delayer* that prevents the player from performing action 2 can be implemented by changing the transition function to ensure that the game will only transition among states in which action 2 cannot be performed (e.g., state C is avoided by switching from τ_1 to τ_2).

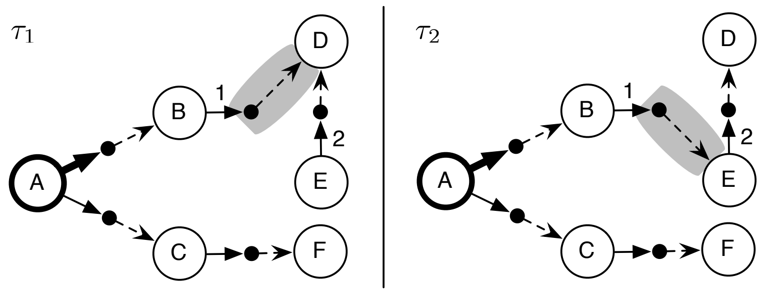


Figure 5.4: Representing a Moe *move substitution* or *future hint* as changing a transition function. When the transition function is changed from τ_1 to τ_2, the player will be forced to observe a hint in state E (a *future hint*) after performing action 1, and must then perform action 2 (a *move substitution*).

In Figure 5.4, a Moe *move substitution* that forces the player to perform action 2 after performing action 1 can be implemented by changing the transition function from τ_1 to τ_2. Similarly, a Moe *future hint* that forces the player to experience state E (which could deliver the desired hint) can be accomplished by the same change.

Whenever Moe performs multiple moves in immediate succession, their effects on the game are applied incrementally before the next player move can be performed. In terms of GEM, if we consider each Moe move to represent a particular, *incremental* change to the game's previous transition function (τ_t), then applying a given sequence of Moe moves as changes to τ_t will result in a new transition function whose activation in the game would have the same effect on the player's experience as the given sequence of Moe moves. Therefore, every sequence of Moe moves can be represented by a unique GEM transition function, τ.

The Player's Experience

GEM histories ($h \in H$) subsume Moe's representation of the player's experience (a sequence of Moe moves and player moves), because every history contains an ordered sequence of the states, actions (i.e., player moves), and transition functions (i.e., sequences of Moe moves) that have occurred in the player's experience thus far.

Legal Player Moves

Whenever Moe needs to simulate a player move while generating potential futures, it retrieves a set of player moves that are legal at the time. This set is constrained by both the plot graph in Figure 5.1 and the sequence of previous Moe moves (*deniers* and *delayers*) that have either occurred during the player's experience or been simulated during a prior step of the generation process. Just as Moe *deniers* and *delayers* make certain player moves illegal by changing the game's transition function, the constraints imposed by a plot graph can be maintained by changing the transition function as well.

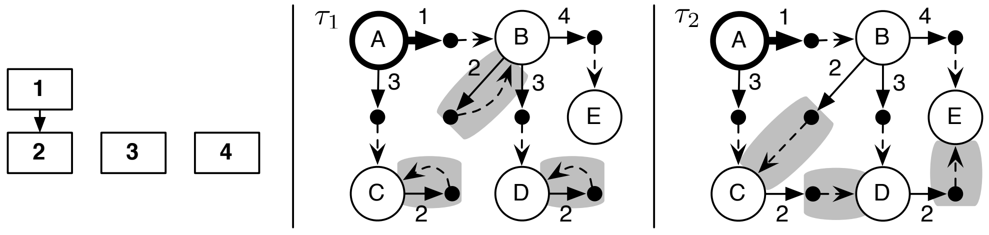


Figure 5.5: Representing a change in which player moves are legal as a change to a transition function. Left: a plot graph that requires action 1 to be performed before action 2. Middle: an MDP with a transition function such that performing action 2 has no effect. Right: an MDP with a transition function such that action 2 can have various effects; this function could be activated after the player performs action 1.

Figure 5.5 shows an example where a plot graph has constrained the player's actions such that action 1 must happen before action 2 (actions 3 and 4 are unconstrained). The transition function τ_1 shows how this constraint can be represented by ensuring that every instance of action 2 always leaves the game's state unchanged[3]. However, as soon as action 1 is performed (say, from state A like in the figure), the manager can change the transition function to τ_2 (or one like it) and thereby allow action 2 to affect the game's state. To ensure that a given action will become available for the player to perform as soon as another action has been performed, the transition functions that are available for selection following the latter action must be constrained such that only those that allow the former action to affect the game's state as desired will be available for selection. I discuss the use of GEM decision constraints below.

The Estimated Player Policy

Moe's estimated player policy (which it uses to compute its objective function) is designed to vary depending on whether or not Moe *hints* or *future hints* have occurred previously in the experience. Given that GEM's estimated player policy is a function of the player's history (which includes transition functions – GEM's analogue to sequences of Moe moves), its output can vary depending on which transition functions have been selected, as Moe requires.

Legal Moe Moves

Before Moe can choose a move to perform, it must determine which of its moves are legal, given the sequence of Moe moves and player moves that have occurred thus far in the experience. Because every sequence of Moe moves can be represented as a single change to a transition function, any *legal* sequence of Moe moves (i.e., where every move is legal with respect to some previous

[3]Ordinarily, I omit such self-edges to reduce clutter when I represent an MDP as a graph, but I show them for action 2 in Figure 5.5 to clarify the present example.

move sequence) can be represented as a single change to a transition function τ that is (implicitly) designer-approved for a given player history. GEM's decision constraint function ($\kappa(h_t)$) is intended to represent precisely this information. To ensure that κ will enforce Weyhrauch's constraints on legal Moe moves, for every history h_t, the set given by $\kappa(h_t)$ can be defined by starting with an empty set and adding one new transition function for every sequence of contiguous Moe moves that can be legally performed following that history.

The Evaluation Function

Moe's evaluation function estimates how successful a given, complete experience will be as a weighted average of several features, each of which is defined in terms of the sequence of Moe moves and player moves that make up the given experience. Moe's features are equivalent to defining a vector of GEM features ($\boldsymbol{f}$) that will only ever be evaluated for complete player histories, and computing its evaluation function amounts to an intermediate step of computing an objective function (ϕ) in GEM. More specifically, Moe's recursive backup procedure for estimating how successful the player's experience will be following a given sequence of Moe moves *is* a GEM objective function: it computes a real number (based on backed-up estimates from the evaluation function) given a sequence of Moe moves (i.e., a transition function) and a sequence of prior moves (i.e., a history), and that number is an estimate of how successful the player's experience will be if those moves are performed following the given sequence of prior moves.

Manager & Policy

Moe's policy considers every complete player experience that could occur as the result of a legal sequence of contiguous Moe moves, and performs the sequence that it expects to yield the most successful experience, given the moves that have already occurred. In terms of GEM, it considers every possible experience that could occur as a result of each of the available, designer-approved transition functions ($\tau \in \kappa(h_t)$), and returns the τ that maximizes its objective function, given the history of the player's experience thus far. Since its operation depends on a set of decision constraints, an objective function, a rollout function, an estimated player policy, and a vector of features (i.e., Blocks #1 through #5), Moe's manager and policy can be represented by Equations 5.1 and 5.2, respectively, where each parameter has been defined in the preceding text:

$$m_{\text{Moe}} = \langle \chi, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f} \rangle \tag{5.1}$$

$$\chi(h_t, e_t, \phi, \kappa, \tilde{\pi}, \boldsymbol{f}) = \operatorname*{arg\,max}_{\tau \in \kappa(h_t)} \phi(\tau, h_t, e_t, \kappa, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{5.2}$$

Table 5.1 summarizes how Moe can be represented using GEM.

Moe Component	GEM Representation
Player Moves	Player Actions
Moe Moves	Changing Transition Functions (both single moves and sequences)
The Player's Experience	Histories
Legal Player Moves	Transition Function & Decision Constraint Function
Estimated Player Policy	Estimated Player Policy
Legal Moe Moves	Decision Constraint Function
Evaluation Function	Features & Objective Function
Manager Policy	Manager Policy (Equation 5.2)

Table 5.1: A summary of how each of Moe's components can be represented using GEM.

5.2 *Left 4 Dead*'s AI Director

Released by Valve Corp. in 2008, *Left 4 Dead* is a commercial, first-person shooter video game in which a team of four survivors (see Figure 5.6) must battle through and ultimately escape from large numbers of zombies (AI-controlled opponents). The player controls the actions of one of the survivors, with the other three being controlled either by other (human) players or by collaborative AI agents. Some of *Left 4 Dead*'s gameplay is managed by an experience manager called "the AI Director" (Booth 2009), which was created to improve the replay value of the game by varying the emotional intensity of the player's experience in a controlled, yet unpredictable way.

Figure 5.6: The four survivor characters in *Left 4 Dead*. Source: (Booth 2009).

5.2.1 The Player's Experience

In *Left 4 Dead*, the player's experience is an alternating sequence of game states and player actions. Examples of actions that players can perform in *Left 4 Dead* include moving among a group of connected rooms, shooting zombies, and gathering items. Since players are not expected to perform one of these actions at every possible opportunity, a timer is used to automatically move the game to a new state if no player action has been taken.

5.2.2 The AI Director's Operations

During gameplay, the AI Director can perform two kinds of operations. First, it can change the current method of *threat population* (i.e., strategies for controlling the composition and creation rate of additional groups of zombies). When *full* threat population is being performed, various kinds of zombies (both minor threats and major threats) are created continually outside of the player's view. When *minimal* threat population is being performed, only minor zombies are created. When *zero* threat population is being performed, no new zombies are created. Regardless of which method is being performed, certain pre-scripted encounters with challenging "boss" zombies may still occur. Second, it can set a *delay timer* to a given value, to delay changing the current threat population method until the given amount of time has elapsed.

5.2.3 The AI Director's Policy

The AI Director was created to support a "powerfully compelling and replayable experience" by causing the player to go through "unpredictable peaks and valleys of intensity" while playing (Booth 2009; pg. 78). To accomplish this task, it repeatedly switches between its available methods of threat population (*full*, *minimal*, and *none*) based on estimates of the intensity of each survivor's emotions. Specifically, for each survivor, the AI Director monitors their experience for particular events that are expected to cause player stress, and thereby increase their *emotional intensity*. Examples of such events include being injured, being incapacitated, or being pulled off a ledge by enemies. Whenever one of these events happens to a survivor, their estimated emotional intensity (represented by a non-negative real number) is increased in proportion to the severity of the event (e.g., being heavily injured results in a larger increase than being lightly injured). Whenever a survivor is not currently fighting an enemy, their estimated emotional intensity is decreased by a small amount (lower-bounded at zero).

When the game begins, the estimated emotional intensity for each survivor is set to zero, and the *full* threat population method is enabled by default. If at any point the survivor team is closer than a given threshold distance to the nearest unvisited safe area, the threat population method is set to *full*. Otherwise, the following behaviour occurs. Whenever the estimated emotional intensity for any of the four survivors rises (due to combat with in-game enemies) above a pre-set *peak threshold*, the AI Director arranges for the threat population method to be set to *zero* following a 3 to 5 second delay. The peak threshold is meant to represent a level of intensity above which players will begin to feel overwhelmed, while the delay is meant to ensure that the active method of threat population remains at *full* for a minimum amount of time (representing one of the desired "peaks" of emotional intensity). The purpose behind changing the method to *zero* (and thereby drastically reducing the number of enemies being generated in the area) is to help ensure that players will never end up feeling persistently overwhelmed. Somewhat later in the game, the estimated emotional intensity of every survivor will have decayed below the peak threshold (because the survivors have destroyed their enemies). As soon as this occurs, the AI Director arranges for the *full* method of threat population to become active following a 30 to 45 second delay. The purpose of this delay period is to ensure that players will have some time to relax before the next wave of zombies attacks (representing one of the desired "valleys" of emotional intensity). In the meantime, if the estimated emotional intensity of every survivor decays below a pre-set *calm threshold*, then the AI director immediately changes the threat population method to *minimal*. I assume that this change occurs to avoid having the survivor team become bored before *full* threat population resumes, while still allowing them some time to relax; Booth gives no explicit reasons for this change (Booth 2009). The net result of the AI Director's policy is that the estimated emotional intensity of each survivor follows a roughly sinusoidal curve over time, as shown in Figure 5.7.

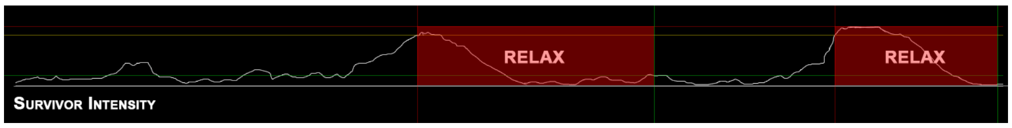


Figure 5.7: A plot of the estimated emotional intensity of one player over time while playing *Left 4 Dead*. Source: (Booth 2009). The red areas show the times during which either *zero* or *minimal* threat population is active; the black areas show times during which *full* threat population is active.

5.2.4 GEM Representation

To demonstrate how the AI Director can be represented in terms of GEM, I will consider each of the AI Director's components in turn.

Player Actions

Player actions in *Left 4 Dead* can be represented directly by GEM actions, provided that an explicit *no-op* action (short for "no operation") occurs whenever the player performs no active action (e.g., as initiated by pushing a button on a game controller) within a certain amount of time.

Threat Population Methods

In terms of GEM, each of the three threat population methods can be well thought of as variants of the game's transition function; that is, while they would vary from one another with respect to how enemies are added into the game, they would be identical with respect to how the rest of the game's dynamics operate (e.g., calculating bullet trajectories, simulating zombie behaviours, or causing survivors to incur damage when they are attacked). The purpose of each threat population method is to give definitive answers two questions:

- "Given the current game state and the player's most recent action, should a new group of zombies be created in the next game state that occurs?", and, if so:

- "Which zombies should comprise this group and where in the environment should they be created?"

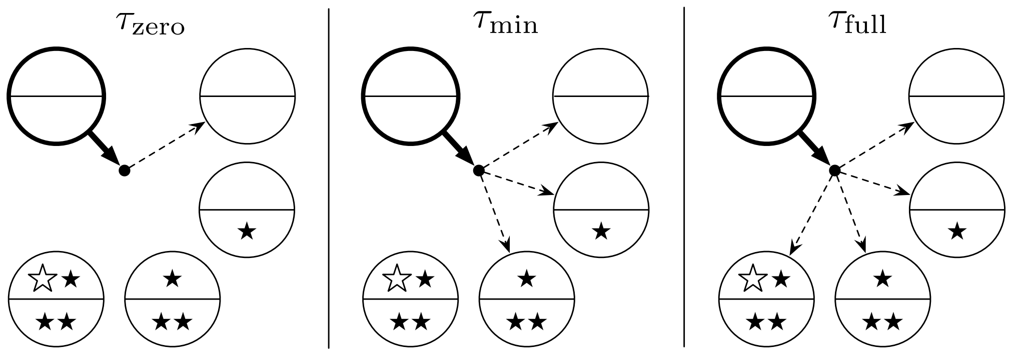


Figure 5.8: Representing each of the AI Director's three threat population methods as a (non-deterministic) transition function. For simplicity, each circle shows the state of only two rooms in the game (separated by a horizontal line), and only four potential subsequent states are shown. Large white / small black stars indicate the presence major/minor zombies. Dashed arrows show the transitions from the most recent state and action (the thick-lined circle and arrow) to potential subsequent states. Small black dots show the point at which the transition function is used to determine the next state. Transitions to states with zero probability of occurring are not shown.

For the *zero* method of threat population, the answer to the first question is always "no": for every possible combination of zombie group composition and placement, the probability of it happening in the next state of the game is zero. Therefore, *zero* threat population can be represented by a transition function where the probability of transitioning to a new state in which new zombies have been created remains at zero (e.g., τ_{zero} in Figure 5.8). Similarly, *minimal* threat population can be represented by a transition function that only assigns positive probabilities to two kinds of states: those in which minor zombies are created, and those in which no new zombies are created (e.g., τ_{min} in Figure 5.8). A transition function representing *full* threat population would be similar to the one for *minimal*, except that positive probabilities would also be assigned to states in which major zombies are created (e.g., τ_{full} in Figure 5.8).

The Player's Experience

The stream of states and actions that makes up the player's experience in *Left 4 Dead* can be represented directly by a GEM history h_t (which includes states, actions, and transition functions).

Estimated Emotional Intensity

Given that the AI Director's estimate of each survivor's emotional intensity is a function of their experience thus far, the maximum of these estimates (which is what the AI Director tracks to change the threat population method) can be well thought of as a GEM feature ($f_{\text{MaxIntensity}}$), as computed from the sequence of states in the game's history.

Distance to Safety

The distance of the survivor team to the nearest unvisited safe area can also be calculated by examining the player's history, meaning that it can also be represented compactly as a GEM feature ($f_{\text{SafetyDistance}}$). The foregoing GEM features are shown as a feature vector in Equation 5.3.

$$\boldsymbol{f} = \langle f_{\text{MaxIntensity}}, f_{\text{SafetyDistance}} \rangle \tag{5.3}$$

Manager & Policy

Using the feature vector given in Equation 5.3, the AI Director's policy can be represented as a GEM manager's policy using a decision constraint function. To define the decision constraint function (κ), consider the following algorithm.

1. If the survivor team is close enough to the nearest unvisited safe area ($f_{\text{SafetyDistance}}$), set $\kappa(h_t, \boldsymbol{f}) = \{\tau_{\text{full}}\}$.
2. If $f_{\text{MaxIntensity}}$ rose past the peak threshold between 3 and 5 seconds ago and only τ_{full} has been used since that happened, then set $\kappa(h_t, \boldsymbol{f}) = \{\tau_{\text{full}}, \tau_{\text{zero}}\}$.
3. If $f_{\text{MaxIntensity}}$ rose past the peak threshold 5 or more seconds ago and only τ_{full} has been used since that happened, then set $\kappa(h_t, \boldsymbol{f}) = \{\tau_{\text{zero}}\}$.
4. If $f_{\text{MaxIntensity}}$ fell past the calm threshold in the last time step and $\tau_t = \tau_{\text{zero}}$, then set $\kappa(h_t, \boldsymbol{f}) = \{\tau_{\text{min}}\}$.
5. If $f_{\text{MaxIntensity}}$ fell past the peak threshold between 30 and 45 seconds ago and only τ_{zero} or τ_{min} have been used since that happened, then set $\kappa(h_t, \boldsymbol{f}) = \{\tau_t, \tau_{\text{full}}\}$.
6. If $f_{\text{MaxIntensity}}$ fell past the peak threshold 45 or more seconds ago and only τ_{zero} or τ_{min} have been used since that happened, then set $\kappa(h_t, \boldsymbol{f}) = \{\tau_{\text{full}}\}$.
7. otherwise; set $\kappa(h_t, \boldsymbol{f}) = \{\tau_t\}$.

Each of the seven lines contains two parts: a statement about the player's history (h_t) and a description of the set of transition functions that the decision constraint function should give whenever a history matching the given statement has occurred (i.e., the set that should be given by $\kappa(h_t, \boldsymbol{f})$ if the given statement about h_t is true). To compute $\kappa(h_t, \boldsymbol{f})$, each line can be visited in order (starting from line 1). If the statement about h_t is true, set $\kappa(h_t, \boldsymbol{f})$ as given on the line. Otherwise, visit the next line. The seven statements provide a complete definition of κ because the sets of histories specified by the given statements have no intersection, and their union is equal to H. Given this definition of κ, the AI Director's policy can be defined as a function that, at each time step, draws a transition function from a designer-defined probability distribution over $\kappa(h_t, \boldsymbol{f})$.[4] This sampling

[4]These distributions were not described in Booth's presentation (2009).

process could be defined as part of an objective function (ϕ), similarly to the example of the $\mathrm{softmax}$ operator that I presented in Section 4.1.2. Since the AI Director's policy depends on a set of decision constraints, an objective function, and a vector of features, the AI Director and its policy can be represented by Equations 5.4 and 5.5, respectively, where each parameter has been defined in the preceding text:

$$m_{\text{AI Director}} = \langle \chi, \kappa, \boldsymbol{f} \rangle \tag{5.4}$$

$$\chi(h_t, e_t, \kappa, \phi, \boldsymbol{f}) = \underset{\tau \in \kappa(h_t, \boldsymbol{f})}{\arg\max}\, \phi(\tau, h_t, e_t, \kappa, \boldsymbol{f}) \tag{5.5}$$

Table 5.2 summarizes how *Left 4 Dead*'s AI Director can be represented using GEM.

AI Director Component	**GEM Representation**
Player Actions	Player Actions
Threat Population	Transition Functions (one per method)
The Player's Experience	Histories
Emotional Intensity	Feature
Distance to Safety	Feature
Manager Policy	Manager Policy, Decision Constraints, & Objective Function

Table 5.2: A summary of how each of the AI Director's components can be represented using GEM.

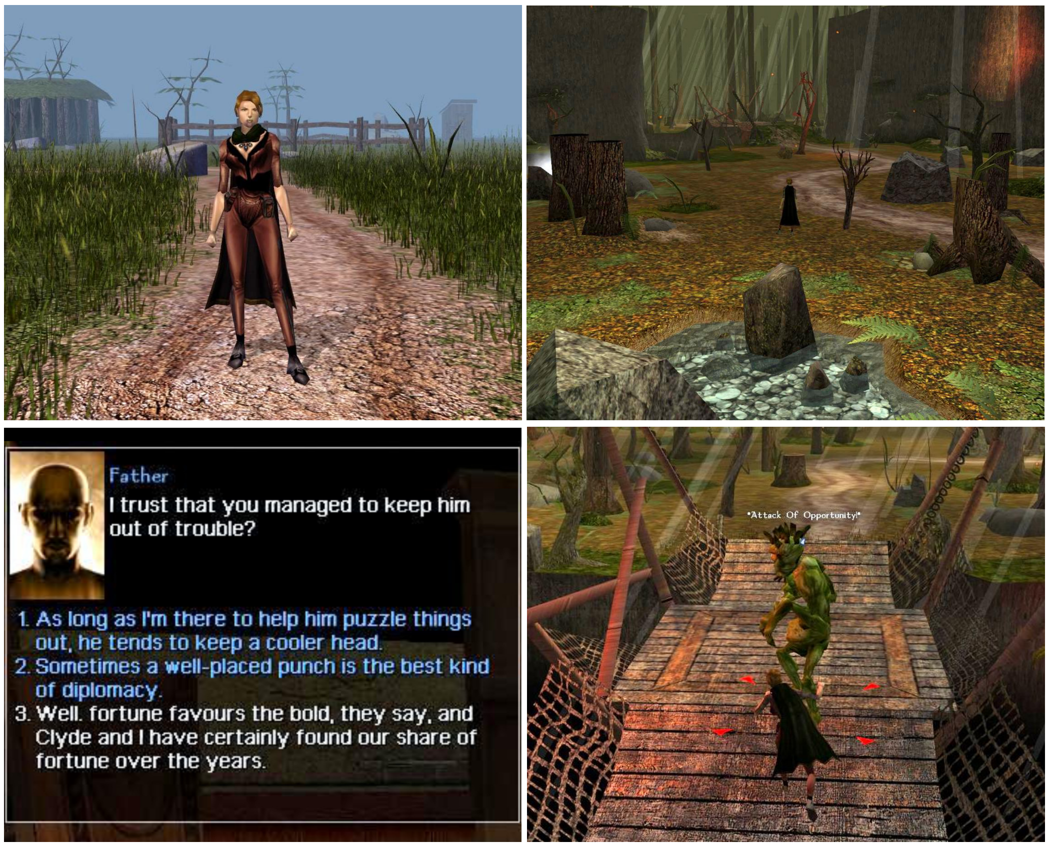
5.3 PaSSAGE

I designed PaSSAGE with the goal of making story-based video games more *fun*, and as a test of my hypothesis that player fun can be increased by dynamically tailoring the content of a game to suit the player's preferences (Thue 2007; Thue et al. 2007a; 2007b; 2008b; 2009; 2010b). PaSSAGE stands for "Player-Specific Stories via Automatically Generated Events".

5.3.1 The Player's Experience

PaSSAGE manages player experiences in the context of single player, story-based video games, where the player controls a story character situated in a virtual environment (this character is often called the player's *avatar*). The player uses their avatar to explore the environment and interact with objects and non-player story characters that they find therein. Figure 5.9 shows screenshots from one such environment, called *Annara's Tale*, which I created as a testbed for PaSSAGE.

The top left image in Figure 5.9 shows the player's avatar: a young woman named Annara. *Annara's Tale* is based loosely on the story of Little Red Riding Hood (Perrault 1697), with Annara replacing the main character of that story (a girl named Red). The top right image in the figure shows part of the environment that players of *Annara's Tale* can explore: Maedorn Forest. *Annara's Tale* contains five areas in total: Annara's home, her village, a tavern in the village, the forest, and a house in the forest where an important story character resides. Every player trajectory through *Annara's Tale* involves Annara travelling to this house to complete an important delivery (similar to


Father

I trust that you managed to keep him out of trouble?

1. As long as I'm there to help him puzzle things out, he tends to keep a cooler head.
2. Sometimes a well-placed punch is the best kind of diplomacy.
3. Well, fortune favours the bold, they say, and Clyde and I have certainly found our share of fortune over the years.

Figure 5.9: Screenshots from *Annara's Tale*. Top left: the player's character, Annara. Top right: Maedorn Forest. Bottom left: dialogue with a non-player character. Bottom right: combat.

Red's delivery to her grandmother's house in Little Red Riding Hood). The bottom left image in the figure shows dialogue – one way in which the player can interact with non-player characters. After reading what the story character says (white text at the top of the image), the player is presented with one or more lines of dialogue for Annara to say as her response (three options are shown in the image), and then the character says a new line or the conversation ends. The bottom right image in the figure shows one of the game's non-player characters: a troll that plays a role similar to the wolf in Little Red Riding Hood. Depending on PaSSAGE's operation, the troll will either attempt to distract Annara from her mission by sending her off the path through the forest, or it will solicit Annara's help in capturing a local wizard (a character similar to the huntsman in Little Red) in exchange for a large sum of money. The bottom right image also demonstrates combat – another one of the ways in which the player can interact with story characters. Choosing to fight the troll (which is analogous to Red fighting the wolf) is one of several deviations from the plot of Little Red Riding Hood that are possible in *Annara's Tale*.

5.3.2 PaSSAGE's Operations

PaSSAGE operates by dynamically choosing between different groups of potential gameplay experiences (i.e., alternating sequences of states and actions that can occur following the player's current history) and then presenting its choice to the player by determining the game's subsequent state.

Encounters

Each group of experiences that PaSSAGE considers is defined by a common story context called an *encounter*. For example, Annara's meeting of the troll and the subsequent combat or dialogue are all part of a single encounter that includes two potential experiences: fighting and killing the troll, or starting and ending a conversation with him.

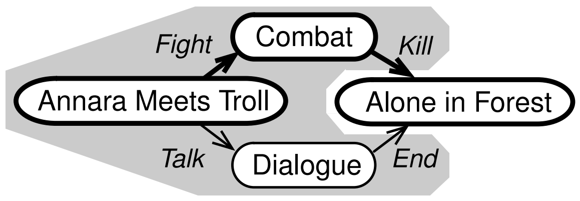


Figure 5.10: An example showing part of my MDP representation of *Annara's Tale*. The shaded region shows a subgraph that makes up the encounter "Annara and the Troll".

In terms of GEM, each encounter can be represented as a collection of the states and actions that the interactive environment (i.e., $\langle S, A, \tau\rangle$, for any $\tau \in \mathcal{T}$). For example, the highlighted region in Figure 5.10 shows the collection of states and actions that represents Annara's encounter with the troll. I say that a transition function τ "causes an encounter to occur" when, given τ, every possible future of the player's current experience will traverse some of the states in the collection that represents that encounter. PaSSAGE's selection of encounters can thus be represented in GEM as changing the transition function of an MDP.

5.3.3 PaSSAGE's Policy

PaSSAGE's objective when choosing encounters is to give the player opportunities to play in whichever *styles of play* they seem to be inclined toward, under the hypothesis that the game will be more fun for them to play as a result. The five styles that PaSSAGE considers are "Fighter" (for players who enjoy combat), "Method Actor" (for players who enjoy complex decisions), "Storyteller" (for players enjoy complex plots), "Tactician" (for players who enjoy thinking creatively), and "Power Gamer" (for players who enjoy finding loot and improving their in-game character). These types were initially proposed by Laws in the context of pen-and-paper Role-Playing Games (Laws 2001), which are similar to many story-based video games.

The Player Model

By observing the player's in-game actions, PaSSAGE automatically estimates a *model* of their preferences toward playing in one the five different styles of play. In the model, the player's inclination toward (or away from) playing in each particular style is represented by a single scalar value; the larger the value, the stronger the inclination.

To allow PaSSAGE to estimate the player model, the story author annotates a subset of the game's state/action pairs with increments that should be applied to the values in the player model, in the event that one of those pairs occurred during the game. The size of the increment represents the author's estimate of how strongly the occurrence of that state/action pair indicates that the player is inclined toward the incremented style of play. For example, I (as the author of *Annara's Tale*) annotated the state/action pair $\langle$Annara Meets Troll, *Fight*$\rangle$ with an increment of 40 for the model's Fighter value. Whenever PaSSAGE observes that state/action pair, it increases its estimate of the player's inclination toward combat (i.e., the Fighter value in the model) by 40 points, representing a "medium" strength indication of the player being a Fighter (e.g., see Figure 5.11).

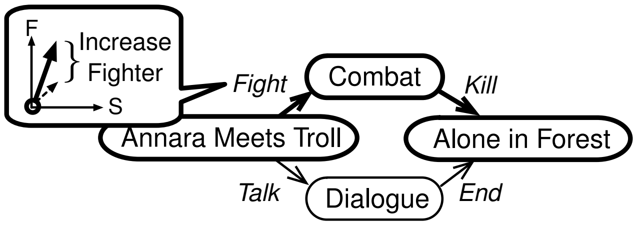


Figure 5.11: A simplified example of learning PaSSAGE's player model in *Annara's Tale*. When the player causes Annara to fight the Troll, the model's value for the player's inclination toward fighting ("F") is increased. Only two play styles are shown here, whereas PaSSAGE uses five.

In terms of GEM, learning the player model can be represented by a vector of five features: $\boldsymbol{f}^{\mathbf{m}} = \langle f^{\mathrm{m}}_{\text{Fighter}}, f^{\mathrm{m}}_{\text{Method Actor}}, f^{\mathrm{m}}_{\text{Storyteller}}, f^{\mathrm{m}}_{\text{Tactician}}, f^{\mathrm{m}}_{\text{Power Gamer}} \rangle$. Each feature examines all of the state/action pairs in a given history and sums up any increments that the author has annotated for the feature's associated play style. For example, computing $f^{\mathrm{m}}_{\text{Fighter}}(h_t)$ with history $h_t =$ (Annara Meets Troll, *Fight*, Combat, *Kill*) would involve summing author annotations for the Fighter play style for each of the two state action pairs that occur in that history: $\langle$Annara Meets Troll, *Fight*$\rangle$: 40 (as described above), and $\langle$Combat, *Kill*$\rangle$: (no annotation[5]), resulting in a final value of 40 for $f^{\mathrm{m}}_{\text{Fighter}}(h_t)$. PaSSAGE's *player model vector* ($\boldsymbol{f}^{\mathbf{m}}$) would thus provide a vector of five scalar values (one for each style of play) representing the player's inclinations toward (or away from) those styles following any given history h_t.

[5]Although it is a fighting-based action, the *Kill* action in Figure 5.11 is not annotated in *Annara's Tale* because the player has no choice but to kill the Troll once combat begins. Whenever the player is forced to perform an action, no play-style preference information can be learned.

Experience Annotations

To successfully match each player's play-style inclinations to different encounters (and the potential experiences therein), PaSSAGE requires another collection of annotations from the story's author. Specifically, each potential experience must be annotated with a vector of five values (one value per play style), where each value represents the author's estimate of how much a player who is inclined toward its corresponding play style will enjoy the activity that is being annotated. For example, I annotated the activity that represents Annara talking with the troll (i.e., (Annara Meets Troll, *Fight*, Combat, *Kill*) in Figure 5.11) with the vector ⟨F:8, M:0, S:0, T:0, P:0⟩. This vector represented my expectation that players who were inclined toward fighting would greatly enjoy fighting the troll (value: 8), and that inclinations toward other play styles would not affect the player's enjoyment of this activity (value: 0).

Similarly to the player model, the author's annotations on potential experiences can be represented as a vector of GEM features: $\boldsymbol{f}^{\mathbf{c}} = \langle f^{\mathrm{c}}_{\text{Fighter}}, f^{\mathrm{c}}_{\text{Method Actor}}, f^{\mathrm{c}}_{\text{Storyteller}}, f^{\mathrm{c}}_{\text{Tactician}}, f^{\mathrm{c}}_{\text{Power Gamer}}\rangle$; I refer to the vector that results from computing these functions for a given trajectory (i.e., from $\boldsymbol{f}^{\mathbf{c}}(h)$) as PaSSAGE's *conditional fun vector*.[6] For any given trajectory h (which represents a potential experience), $\boldsymbol{f}^{\mathbf{c}}(h)$ retrieves the annotation vector that the author created for that trajectory (if any); if no vector was defined, $\boldsymbol{f}^{\mathbf{c}}(h)$ returns a zero vector (i.e., ⟨0, 0, 0, 0, 0⟩).

Encounter Selection

Whenever an opportunity arises to choose which encounter should happen next (and thereby which set of activities will become available), PaSSAGE examines each encounter as follows. For each activity in the encounter, it retrieves that activity's annotation vector and computes its inner product with a vector representation of the player model (the same order of play styles is used for both vectors); the computed value represents how much fun that activity is expected to be. The activity that corresponds to the highest expected fun value over all of the available encounters is identified as the best available activity, and the encounter that contains it is then chosen to occur next in the story (ties are broken randomly). The set of encounters that become available for PaSSAGE to select changes as the player's experience proceeds, as decided by the author of the game being managed.

5.3.4 GEM Representation

In terms of GEM, encounter selection (i.e., changing the game's transition function to cause encounters to occur) is performed by PaSSAGE's policy (χ). Experience management can be performed

[6] I use the term "conditional" when describing this vector's fun values for the following reason. For each play style that is represented in the vector, the author estimates how much fun the player would derive from playing through a given experience *under the assumption* that the player generally enjoys playing in that style. For example, $f^{\mathrm{c}}_{\text{Fighter}}(h)$ only estimates the degree to which a player who is inclined toward fighting will enjoy playing through history h – it does not estimate the enjoyment of players who have other inclinations.

with PaSSAGE using all five GEM building blocks, as shown in Equation 5.6:

$$\chi(h_t, e_t, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}) = \underset{\tau \in \kappa(h_t)}{\arg\max}\, \phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{5.6}$$

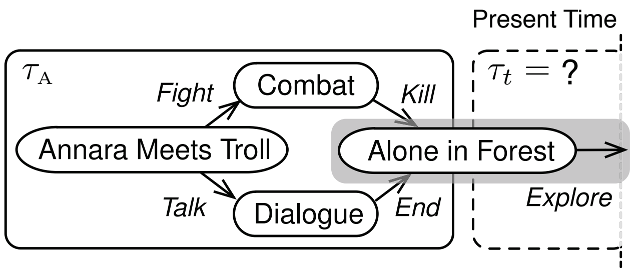


Figure 5.12: Part of *Annara's Tale* represented as an MDP. The shaded area marks a decision point. The solid-edged box shows the MDP for time step $t-1$ and earlier (i.e., $\langle S, A, \tau_{\text{A}}\rangle$), and the dashed-edged box shows that a transition function must be selected, to be used as τ_t.

Using a decision constraint function (κ; Block #1), the author can ensure that only certain encounters are available for selection following particular player histories. This guarantee can be achieved by constraining the sets of transition functions that are available in such a way that only those that each cause a desired encounter to occur are available for the manager to select. For example, following the decision point $\langle$Alone in Forest, *Explore*$\rangle$ shown in Figure 5.12, I (as author of *Annara's Tale*) specified that only two transition functions should be available for the manager to select: one that causes an encounter where Annara meets a puzzled traveller (τ_{A}), and one that causes an encounter where giant spiders attack (τ_{B}); that is, $\kappa(h_t) = \{\tau_{\text{A}}, \tau_{\text{B}}\}$ for every history h_t that ends in $\langle$Alone in Forest, *Explore*$\rangle$. Figure 5.13 shows each of these encounters in the context of the MDP that would be created if τ_{A} or τ_{B} were selected, respectively.

For each available transition function, an objective function (ϕ; Block #2) can be used to estimate the player's return. Given my goal for PaSSAGE, the player's return is how much fun the player will derive from their experience in the game. PaSSAGE's objective function estimates this value as shown in Equation 5.7.

$$\phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = \sum_{h^+ \in \rho(s_t, a_t, \tau, e_t)} \Pr(h^+ | h_t, \chi, e_t, \tilde{\pi})\big(\boldsymbol{f}^{\mathbf{m}}(h_t) \cdot \boldsymbol{f}^{\mathbf{c}}(h^+)\big) \tag{5.7}$$

The objective function begins by performing rollouts of the game's future (using a rollout function, ρ; Block #3) from the current decision point ($\langle s_t, a_t\rangle$) onward. PaSSAGE's rollout function collects *bounded* rollouts – that is, rollouts that do not necessarily extend to an ending of the game. PaSSAGE's rollouts are bounded by decision points; if extending a trajectory h^+ during rollout generation would cause it to include a decision point or an ending of the game, then that trajectory is extended no further. The result is that each rollout $h^+ \in \rho(s_t, a_t, \tau_t, e_t)$ stops with the first transition function that immediately precedes either a decision point or an ending of the game. PaSSAGE's rollouts are restricted in this way because the potential for player preferences to change

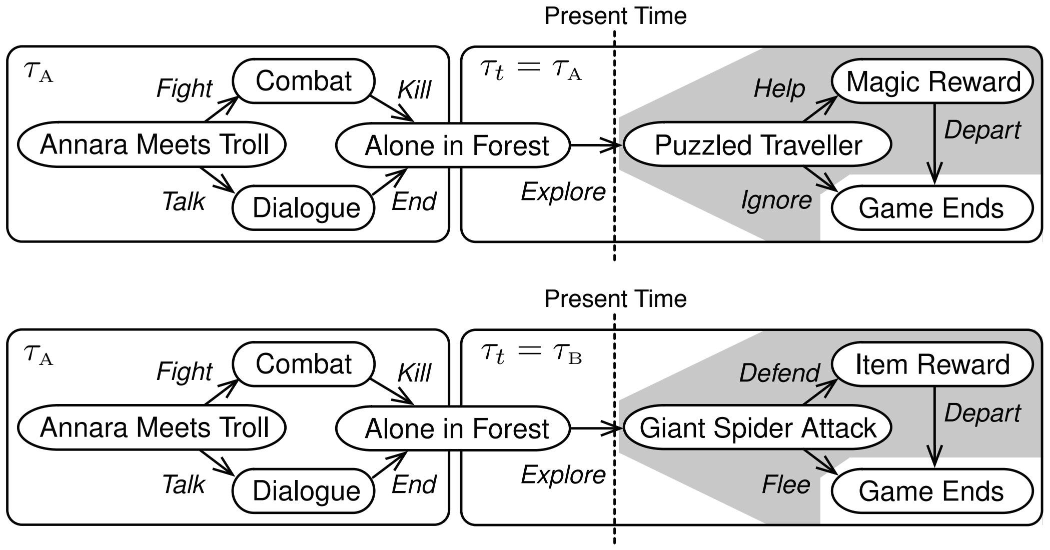


Figure 5.13: Part of *Annara's Tale* represented as an MDP, showing the effects of choosing one of two available transition functions (τ_A or τ_B) with respect to which encounter will occur next. Ovals are game states, arrows show the combined effect of a single player action and a deterministic transition function. Each shaded area shows a collection of states and actions that represents a unique encounter. Top: The MDP if τ_A were selected. Bottom: The MDP if τ_B were selected.

during the game decreases the value of any long-term planning. Formally, for every state/action pair $\langle s_t, a_t\rangle \in S \times A$, if $\langle s_t, a_t\rangle$ is *not* a decision point, then for all histories $h_t \in H$ that end with $\langle s_t, a_t\rangle$, $\kappa(h_t) = \{\tau_t\}$.

The feature vector ($\boldsymbol{f}$; Block #5) in Equation 5.7 is obtained by concatenating $\boldsymbol{f}^{\mathbf{m}}$ and $\boldsymbol{f}^{\mathbf{c}}$. For each generated rollout h^+, PaSSAGE computes an inner product between the conditional fun vector of that rollout ($\boldsymbol{f}^{\mathbf{c}}(h^+)$) and the current player model vector ($\boldsymbol{f}^{\mathbf{m}}(h_t)$). Intuitively, the better the match between the player model and the conditional fun of the given rollout, the more fun the player will derive from playing through that rollout. PaSSAGE approximates the probability term in Equation 5.7 (which includes an estimated player's policy, $\tilde{\pi}$; Block #4) by assigning a probability of 1 to the single rollout that maximizes the inner product between $\boldsymbol{f}^{\mathbf{m}}(h_t)$ and $\boldsymbol{f}^{\mathbf{c}}(h^+)$. As a result of this approximation, Equation 5.7 becomes Equation 5.8:

$$\phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = \max_{h^+ \in \rho(s_t, a_t, \tau, e_t)} \left(\boldsymbol{f}^{\mathbf{m}}(h_t) \cdot \boldsymbol{f}^{\mathbf{c}}(h^+)\right) \tag{5.8}$$

This approximation is the result of assuming that the player will always perform whichever actions seem to be the most fun to them at the time. The rollout that maximizes the inner product in Equation 5.8 represents PaSSAGE's best guess at what those actions might be.

Example

The following example demonstrates how PaSSAGE would choose between two transition functions in *Annara's Tale*. Figure 5.12 shows one of the game's decision points ($\langle s_t, a_t\rangle = \langle$Alone in Forest, *Explore*$\rangle$) and two player histories that could lead to its occurrence in-game (one where Annara fights a troll, and one where she talks to it instead). As indicated by the term τ_t in the figure, PaSSAGE must select a transition function to use from the current time step (t) until either a new decision point is reached or the game ends. In this example, I demonstrate what PaSSAGE would do following the given decision point for two different players: one who fought and killed the troll (P1), and one who talked to it instead (P2).[7] Player 1's history is thus: $h_{t,\text{P1}} = ($Annara Meets Troll, *Fight*, τ_A, Combat, *Kill*, τ_A, Alone in Forest, *Explore*$)$ while player 2's history is $h_{t,\text{P2}} = ($Annara Meets Troll, *Talk*, τ_A, Dialogue, *End*, τ_A, Alone in Forest, *Explore*$)$.

When given either of these histories, the decision constraint function for *Annara's Tale* (κ) gives the same set of two transition functions for a manager to choose from (i.e., $\kappa(h_{t,\text{P1}}) = \kappa(h_{t,\text{P2}}) = \{\tau_\text{A}, \tau_\text{B}\}$). Figure 5.13 shows how the MDP would look if either τ_A (top) or τ_B (bottom) were chosen; τ_A leads to Annara encountering a puzzled traveller who will give her a magical reward if she helps, and τ_B leads to Annara witnessing an attack by giant spiders, whose victim will give her an item if she comes to their defense. To decide which of these transition functions to select, PaSSAGE uses each one to generate a set of bounded rollouts from $\langle s_t, a_t\rangle$; each rollout stops with the first transition function that immediately precedes either a decision point or an ending of the game. The third column of Table 5.3 shows the four rollouts that PaSSAGE would generate using τ_A and τ_B following the decision point $\langle$Alone in Forest, *Explore*$\rangle$. The second column of Table 5.3 gives a short label to each rollout in the second column, to simplify cross-referencing Tables 5.3 and 5.4.

τ		Rollout (h^+)	Cond. Fun ($\boldsymbol{f}^\mathbf{c}(h^+)$)
τ_A	A1	(Puzzled Traveller, *Help*, τ_A, Magic Reward, *Depart*)	$\langle$F:0, M:0, S:4, T:8, P:2$\rangle$
	A2	(Puzzled Traveller, *Ignore*)	$\langle$F:0, M:0, S:2, T:0, P:0$\rangle$
τ_B	B1	(Giant Spider Attack, *Defend*, τ_B, Item Reward, *Depart*)	$\langle$F:8, M:4, S:2, T:0, P:2$\rangle$
	B2	(Giant Spider Attack, *Flee*)	$\langle$F:0, M:4, S:0, T:0, P:0$\rangle$

Table 5.3: Rollouts that PaSSAGE would generate for each to two available transition functions at the decision point shown in Figure 5.12. The conditional fun vector for each rollout is also shown. The symbols in column 2 facilitate cross-referencing rollouts with Table 5.4.

For each generated rollout h^+, PaSSAGE computes the inner product shown in Equation 5.8. Values for the conditional fun vector of each rollout ($\boldsymbol{f}^\mathbf{c}(h^+)$) are given in the fourth column of Table 5.3, and repeated for convenience in the third column of Table 5.4. Assumed models for players 1 and 2 ($\boldsymbol{f}^\mathbf{m}(h_{t,\text{P1}})$ and $\boldsymbol{f}^\mathbf{m}(h_{t,\text{P2}})$, respectively) are shown at the top of the last two columns of Table 5.4. To simplify this example, I have assumed that each player's history has lead PaSSAGE to estimate (via $\boldsymbol{f}^\mathbf{m}$) that player 1 plays like a Fighter (F:100) and a Tactician (T:40), and that player 2 plays like a Method Actor (M:40) and a Storyteller (S:100).

Columns 4 and 5 show the value of the inner product that PaSSAGE calculates for each rollout (for players 1 and 2, respectively). Italic values indicate the rollout that maximizes the inner product

[7]To simplify this example, I ignore the details of earlier parts of *Annara's Tale* and instead assume that each player's history was such that it would have resulted in the model vector that is shown for that player in Table 5.4.

			Player 1 (P1)	Player 2 (P2)
			Model ($\boldsymbol{f}^{\mathbf{m}}(h_{t,\mathrm{P1}})$) ⟨F:100, M:0, S:0, T:40, P:0⟩	Model ($\boldsymbol{f}^{\mathbf{m}}(h_{t,\mathrm{P2}})$) ⟨F:0, M:40, S:100, T:0, P:0⟩
τ	h^+	Cond. Fun ($\boldsymbol{f}^{\mathbf{c}}(h^+)$)	Inner Product	Inner Product
τ_A	A1	⟨F:0, M:0, S:4, T:8, P:2⟩	*320*	***400***
	A2	⟨F:0, M:0, S:2, T:0, P:0⟩	0	200
τ_B	B1	⟨F:8, M:4, S:2, T:0, P:2⟩	***800***	*360*
	B2	⟨F:0, M:4, S:0, T:0, P:0⟩	0	160

Table 5.4: Using the models of two different players to estimate their return for each of two candidate transition functions. Italics show the highest estimated return over all available transition functions. Bold values indicate the highest estimated return over all possible rollouts. The symbols in the rollouts column (h^+) can be cross-referenced with Table 5.3.

calculation for each transition function, and the maximums of these values across both of the available transition functions are shown in bold. The bold value for each player indicates which transition function PaSSAGE would expect to maximize their return (i.e., τ_B for player 1, and τ_A for player 2). Equations 5.6 and 5.8 would thus lead PaSSAGE to select τ_B if player 1 was playing (leading to the spider attack), and τ_A if player 2 was playing instead (leading to the puzzled traveller).

Table 5.5 summarizes how PaSSAGE can be represented using GEM.

PaSSAGE Component	**GEM Representation**
Player Actions	Player Actions
Probability of Player Actions	Estimated Player Policy (Block #4)
The Player's Experience	Histories
Gameplay Activities	Trajectories
Encounters	Groups of Trajectories
Player Model	Features (Block #5)
Gameplay Activity Annotations	Features (Block #5)
Bounded Rollouts	Rollout Function (Block #3)
Encounter Selection	Manager Policy, Decision Constraints (Block #1), & Objective Function (Block #2)

Table 5.5: A summary of how each of the PaSSAGE's components can be represented using GEM.

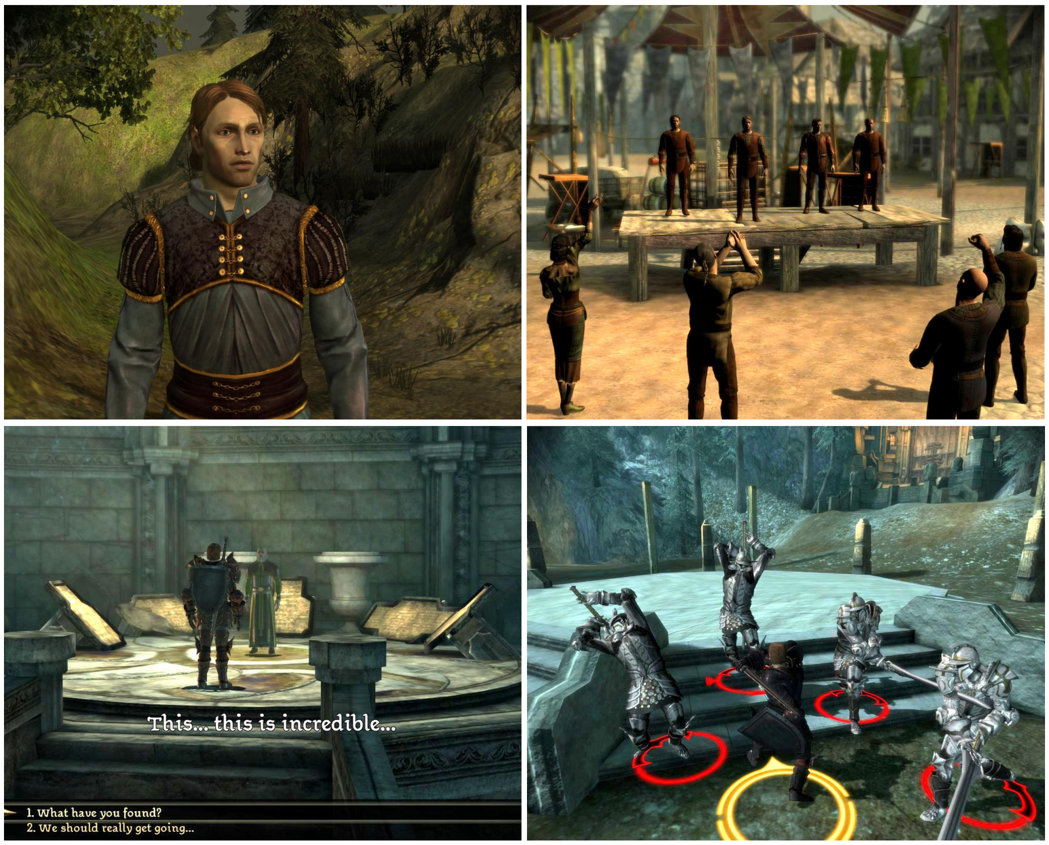
5.4 PaSSAGE 2

I created PaSSAGE 2 (Thue et al. 2010a; 2011) to investigate whether the dynamically learned model of gameplay styles that we developed for PaSSAGE could be used to strengthen players' beliefs that they have control over their experience in a game. This type of control is commonly referred to as *agency* (Wardrip-Fruin et al. 2009).

The belief that one's actions have an effect on the world (i.e., that one has agency) is a fundamental craving of human nature, having been linked to emotional well-being, improved performance, and good health (Bandura 2001; Thompson and Spacapan 1991). While many commercial video games attempt to instil and strengthen this belief in players by giving them many opportunities to affect the game's world (e.g., *Grand Theft Auto V* (Rockstar North 2013) or *Skyrim* (Bethesda Game Studios 2011)), Thompson et al. found that people's judgements of their own agency depend on other factors as well (Thompson et al. 1998). More specifically, they identified two primary factors: (a) the connection that one perceives between their action and the outcome that occurred as a result, and (b) one's intention to achieve that outcome. They further divided intentionality into three subfactors: (i) one's ability to cause an outcome to occur, (ii) how easy the outcome was to foresee, and (iii) the outcome's desirability. They called this work "The Control Heuristic". Based on Thompson et al.'s (1998) finding that lowering any one of these factors could weaken one's belief that they have agency, I hypothesized that such beliefs might be *strengthened* in video games by ensuring that particular player choices in-game always result in *desirable* outcomes (factor b.iii).

5.4.1 The Player's Experience

Like PaSSAGE, PaSSAGE 2 manages player experiences in the context of single player, story-based video games. Figure 5.14 shows some screenshots from *Lord of the Borderlands*, the game that I created as a testbed for PaSSAGE 2.


This... this is incredible...

1. What have you found?
2. We should really get going...

Figure 5.14: Screenshots from *Lord of the Borderlands*. Top left: the player's avatar, Jaden. Top right: Jaden's village. Bottom left: dialogue with a non-player character. Bottom right: combat.

The top left image in Figure 5.14 shows the player's avatar: a young man named Jaden. *Lord of the Borderlands* is an original interactive story (co-written by me, Charles Crittenden, and Trevon Romanuik) in which Jaden progresses from being a student at a military training academy to gaining authority over the lands in which he lives (the Borderlands). Although every player's story follows this general progression, the means by which Jaden gains his authority and the state of the story's world when the game ends can vary substantially from one play-through to the next. For example, the top right image in the figure shows a scene from one of the game's 16 different endings, where the player is running as a candidate for leadership over the Borderlands in a democratic election. This image also shows Jaden's village – one of the six areas that players can explore in *Lord of the Borderlands*. The others areas were: a military academy, the road between the academy and Jaden's village, Jaden's family manor near the village, shipping docks near the village, a camp held by the military, and a remote site with ancient, underground ruins. The bottom left image shows a scene just prior to two endings that can happen in the ancient ruins: the player has discovered information that could topple the empire that controls the Borderlands, and must choose to either release it in support of a rebellion against the empire, or destroy it in exchange for an empire-granted position as Lord of the Borderlands. This image also shows Jaden in conversation with an important story character: an ex-Imperial wizard who works to expose corruption within the empire. The bottom right image shows an example of combat that can occur outside the ancient ruins, where the player (having previously supported the rebels) must fight through a group of of the empire's soldiers to gain access to the ruins.

Player Choices

In the context of this work, a *player choice* is any state $c \in C \subseteq S$ from which some of the player's actions, if performed, would prevent them from ever reaching one or more of the game's decision points. Intuitively, player choices are similar to the so-called "branching points" of interactive stories, where any state from which the player's actions could lead to different future trajectories is thought of as starting new "branches" in a graph tree representation of the story. I define player choices in terms of decision points (rather than general trajectories) to distinguish them from the player's opportunities to influence PaSSAGE 2's player model; actions that influence the model always lead to different future trajectories, but they might not always lead to different sets of decision points. For example, the beginning of *Lord of the Borderlands* requires the player to choose between several actions that help PaSSAGE 2 estimate the player's model, but every resulting trajectory leads to the state "Rebel Uprising" in Figure 5.15. The state "Rebel Uprising", on the other hand, is a player choice: depending on whether the player decides to join or oppose the rebels, it will only be possible for one of two decision points (shaded in the figure) to occur in the future of that player's experience: either Jaden will return home having gained the favour of the rebels, or he will return home having gained the favour of the empire instead.

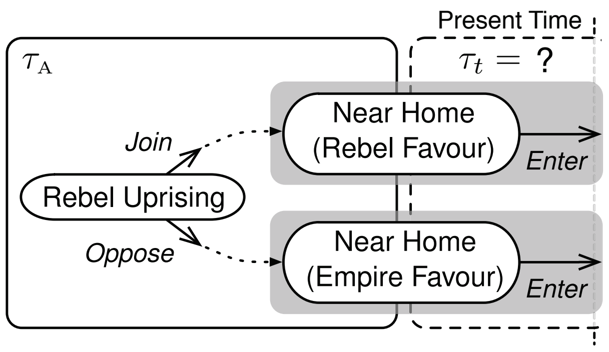


Figure 5.15: Part of *Lord of the Borderlands* represented as an MDP. The shaded areas mark decision points that are potential outcomes of the player choice "Rebel Uprising". The solid- and dashed-edged boxes are defined as in Figure 5.12. The outcome that occurs will vary with the player's choice to "Join" or "Oppose" the rebels. The dashed arrows represent parts of the MDP that I have omitted to simplify the current example.

5.4.2 PaSSAGE 2's Operations

PaSSAGE 2 operates by dynamically determining which potential experiences will be available immediately after the outcome of every player choice.

Outcomes

Given any player choice c, there are some decision points in the game whose occurrence depends on the player's action at that choice. For example, the decision point ⟨Near Home (Rebel Favour)⟩ can only occur if the player chooses to join the rebels at the choice shown in Figure 5.15 ("Rebel Uprising"). I refer to these decision points as the potential *outcomes* of that choice. Formally, an outcome $o \in O \subseteq D$ is a decision point that can only be reached following exactly one player action at some player choice c (regardless of which transition functions might be selected by the manager). Figure 5.15 shows two outcomes of the player choice "Rebel Uprising": the outcome ⟨Near Home (Rebel Favour), *Enter*⟩ is a result of the *Join* action, and the outcome ⟨Near Home (Empire Favour), *Enter*⟩ is a result of the *Oppose* action.

Encounters

Similarly to PaSSAGE, PaSSAGE 2 assumes that every decision point will always be followed by an opportunity to select between different groups of potential experiences called *encounters* (recall Figure 5.10 for an example of an encounter in *Annara's Tale*). Since outcomes *are* decision points, this means that PaSSAGE 2 operates as follows: after the player chooses an action to perform at a given player choice (e.g., choosing to *Join* the rebels in Figure 5.15), an outcome of that choice will eventually occur (e.g., ⟨Near Home (Rebel Favour), *Enter*⟩ in the figure) and PaSSAGE 2 will then be invoked to decide which encounter should immediately follow that outcome.

5.4.3 PaSSAGE 2's Policy

PaSSAGE 2 assumes that the desirability of a video game's content is positively correlated with how much fun it is to play. Under this assumption, the Control Heuristic suggests a way to use the player modelling technique from PaSSAGE to try to strengthen the player's belief that they have agency. Specifically, since the outcomes of player choices are represented as decision points, the player model and potential experience annotations from PaSSAGE can be used to estimate how much fun the player will derive from each of the encounters that can follow any given outcome. Given these estimates, PaSSAGE 2 can then construct an estimate of the desirability of each outcome/encounter pair, toward ensuring that every player choice results in an outcome (and subsequent encounter) that is maximally desirable among those that can occur. According to the Control Heuristic, the player's belief in having agency should then be strengthened as a result.

PaSSAGE 2's use of choices and their outcomes highlights an important distinction between it and PaSSAGE concerning how their intended effects and the techniques that they use affect their decision constraints. With PaSSAGE, decision constraints are unrestricted; any state/action pair that can occur in a given game can be used as a decision point. PaSSAGE can be flexible in this regard because it aims to maximize player fun, and its method for doing so (matching play-styles) does not require players to recognize the connection between their actions and the experiences that occur thereafter (i.e., no connections between choices and outcomes are required). Furthermore, since PaSSAGE's player model is not ordinarily visible to players[8], there is no practical way for players be sure that one of their model-updating actions was also the cause of some later part of their experience (even though it could have been). Unlike PaSSAGE's decision constraints, PaSSAGE 2's decision constraints must be such that every decision point is an outcome of some prior player choice. This restriction is necessary for two reasons: because (i) PaSSAGE 2 uses the Control Heuristic to strengthen players' beliefs in their own agency, and because (ii) the notion of an outcome in the Control Heuristic requires that it be at least practically *possible* for players to recognize a connection between their actions (player choices) and the experiences that occur thereafter (which follow outcomes). Without this perceivable connection, one would have little reason to believe that maximizing the desirability of the encounter that follows an "outcome" (i.e., a decision point with no connected player choice) would strengthen the player's beliefs that they have agency.

5.4.4 GEM Representation

Similarly to PaSSAGE, managing experiences with PaSSAGE 2 can be represented as changing the transition function of an MDP where all five GEM building blocks are used. Except for the differences that I explain in this section, identical representations can be assumed for PaSSAGE and PaSSAGE 2.

Whenever an outcome $o_t = \langle s_t, a_t \rangle$ is reached during gameplay, PaSSAGE 2 uses an objective

[8]A model viewer can be enabled for demonstration purposes.

function (Block #2) to estimate the the player's return (i.e., the strength of their belief in having agency) with respect to each transition function τ that is available for selection (i.e., for each $\tau \in \kappa(h_t)$; recall that every outcome $o \in O$ is a decision point). Following any given outcome, each available transition function causes a different encounter to occur. For example, Figure 5.16 shows the two encounters that could occur following the outcome ⟨Near Home (Rebel Favour), *Enter*⟩ in *Lord of the Borderlands*. In one encounter ("Imprisoned", in the figure), a party is being held by Jaden's uncle to welcome Jaden home, but when the uncle (who supports the empire) learns that Jaden joined in the rebellion at the academy, he throws Jaden in prison; the player then controls Jaden during his escape. In the other encounter ("Friendly Rebels", in the figure), no party is being held – instead, rebels have taken over Jaden's manor, and welcome him warmly in recognition of his helping them at the academy; the player is then invited to help the rebels defend against an approaching imperial army.

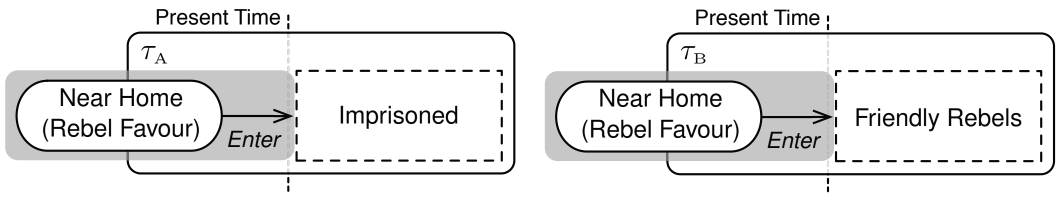


Figure 5.16: Part of *Lord of the Borderlands* represented as an MDP, showing the effects of choosing one of two available transition functions (τ_A or τ_B) at the highlighted outcome. Left: the MDP if τ_A were selected. Right: the MDP if τ_B were selected. Dashed-edge boxes denote encounters whose details I have omitted to simplify the example.

To choose between transition functions, PaSSAGE 2 computes its objective function as a linear combination of two estimates of desirability: the outcome's *local desirability* and its *contextual desirability*.

Local Desirability

For each available transition function τ at a given outcome o, PaSSAGE 2 estimates the outcome's *local desirability* (des_L) similarly to how PaSSAGE estimates the player's return (which represents how much fun the player will have) following a given player history ($des_L : O \times \mathcal{T} \times H \times E \to \mathbb{R}$). Equation 5.9 shows this calculation as a modified version of PaSSAGE's objective function (Equation 5.8), where the first two terms of the rollout function ρ have been changed from s_t, a_t (the current state and action) to o (any given outcome):

$$des_L(o, \tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = \max_{h^+ \in \rho(o,\tau,e_t)} \left(\boldsymbol{f}^{\mathbf{m}}(h_t) \cdot \boldsymbol{f}^{\mathbf{c}}(h^+)\right) \tag{5.9}$$

Contextual Desirability

PaSSAGE 2 uses the *contextual desirability* (des_C) of an outcome o to model how the player might consider the outcomes that could have happened *instead* of o, if had they chosen a different action in response to a prior player choice (e.g., if the player had chosen to help the rebels instead of opposing them in Figure 5.15). Specifically, if the player happened to realize that an alternative to o would have had a higher (local) desirability than o, then the contextual desirability of o would be negative to model a sense of regret. Similarly, it would be positive if the player realized that an alternative to o would have been substantially *less* desirable than o (to model a sense of relief).

To estimate the contextual desirability of an outcome o, PaSSAGE 2 requires an *awareness function* (α_τ). This function estimates the probability that the player will realize, given the trajectories that they could experience following o using transition function τ, that they could have reached a different outcome o' if they had acted differently at a prior player choice (i.e., $\alpha_\tau(o'|o) \in [0,1] \subset \mathbb{R}$).

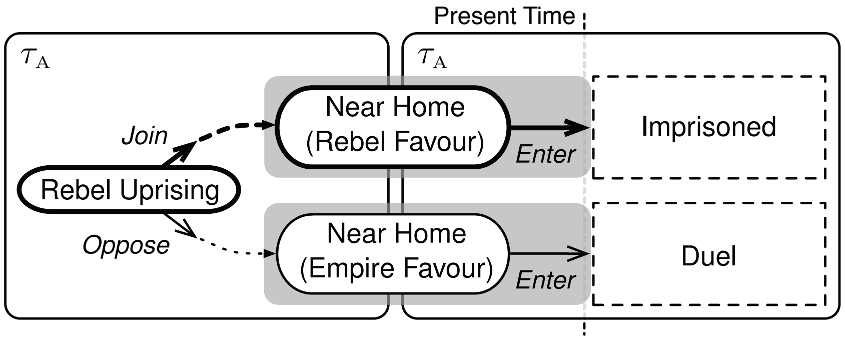


Figure 5.17: Part of *Lord of the Borderlands* represented as an MDP, showing the potential experiences that could follow after the two highlighted outcomes, supposing that PaSSAGE 2 chooses τ_A at each outcome. The bold path shows the current player's history. Dashed-edge boxes denote encounters whose details I have omitted to simplify the current example.

For example, consider the trajectories that the player could experience following player choice "Rebel Uprising" (Figure 5.17). If the player chooses to join the rebels and PaSSAGE 2 selects τ_A when the outcome $o_R = \langle$Near Home (Rebel Favour), *Enter*$\rangle$ is reached, then every following trajectory will involve Jaden being thrown into prison for having helped the rebels at the academy ("Imprisoned" in the figure). In prison, he meets a rebel who says that he had been expecting to fight Jaden in a duel. If the player had instead chosen to oppose the rebel uprising, and PaSSAGE 2 had still selected τ_A following the outcome $o_E = \langle$Near Home (Empire Favour), *Enter*$\rangle$, then every following trajectory would have involved Jaden actually fighting in the aforementioned duel ("Duel" in the figure). Because the imprisoned rebel (after outcome o_R) directly describes the trajectories that could follow o_E (fighting the duel), the awareness function for *Lord of the Borderlands* gives a high probability that the player would be aware of o_E if they experienced o_R ($\alpha_{\tau_A}(o_E|o_R) = 0.8$). For notational convenience, let $\mathcal{A}_{o,\tau} \subseteq O$ describe the set of all outcomes $o' \neq o$ for which $\alpha_\tau(o'|o) > 0$;

$\mathcal{A}_{o,\tau}$ then contains all of the *alternative outcomes* of o given τ. Equation 5.10 shows how PaSSAGE 2 uses the awareness function to estimate the contextual desirability of a given outcome o. For clarity, I use ellipses to abbreviate the required parameters of the local desirability function (des_{L}).

$$des_{\text{C}}(o, \tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = \sum_{o' \in \mathcal{A}_{o,\tau}} \frac{\alpha_\tau(o'|o)}{|\mathcal{A}_{o,\tau}|} \Big(des_{\text{L}}(o, \dots) - des_{\text{L}}(o', \dots) \Big) \tag{5.10}$$

In terms of GEM, PaSSAGE 2's awareness function ($\alpha_\tau(o'|o)$) can represented by a vector of features that contains one feature for each outcome $o' \in O$. Each feature $f^\alpha_{o'}$ in this *awareness vector* ($\boldsymbol{f}^\alpha$) is a function that accepts a trajectory h built from $o = \langle s, a \rangle$ and τ (i.e., $h = (s, a, \tau)$) and gives the probability of the player being aware of o' if they experienced o, assuming that transition function τ was chosen by the manager following both outcomes (i.e., $\forall o' \in O, f^\alpha_{o'} : H \rightarrow \mathbb{R} \in [0, 1]$).

Objective Function & Policy

To balance between its estimates of local and contextual desirability, PaSSAGE 2's objective function is defined as shown in Equation 5.11, where $w \in \mathbb{R}^+$ is a weight that allows the designer to prefer either local or contextual desirability as being more important to computing the overall desirability.[9]

$$\phi(o, \tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) = des_{\text{L}}(o, \dots) + w des_{\text{C}}(o, \dots) \tag{5.11}$$

PaSSAGE 2's policy is thus given by Equation 5.12, where o_t is the last state/action pair in h_t.

$$\chi(h_t, e_t, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}) = \operatorname*{arg\,max}_{\tau \in \kappa(h_t)} \phi(o_t, \tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{5.12}$$

As I described in Section 5.4.3, PaSSAGE 2's decision constraints ensure that every decision point $\langle s, a \rangle$ in the given environment is an outcome of a prior player choice (i.e., $O = D$).

Example

Figure 5.18 demonstrates how the partial MDP from Figure 5.15 could be extended using two different transition functions (τ_{A}, and τ_{B}). Figure 5.19 shows screenshots of the encounters described in Figure 5.18. If the player arrives home with the rebels' favour, selecting τ_{A} will result in Jaden being imprisoned by his uncle and trying to escape ("Imprisoned" in the figures); selecting τ_{B} will result in him finding that rebels have seized his home and need his help to defend it from the empire ("Friendly Rebels" in the figures). Alternatively, if the player arrives home with the *empire's* favour (because they opposed the rebels during the uprising), selecting τ_{A} will result in Jaden fighting a duel against a captured rebel ("Duel" in the figures); selecting τ_{B} will result in him having to remove a group of rebels that has seized his home ("Hostile Rebels" in the figures). Because PaSSAGE 2

[9]For all of my tests of PaSSAGE 2 using *Lord of the Borderlands*, I set $w = 1$ to balance evenly between local and contextual desirability.

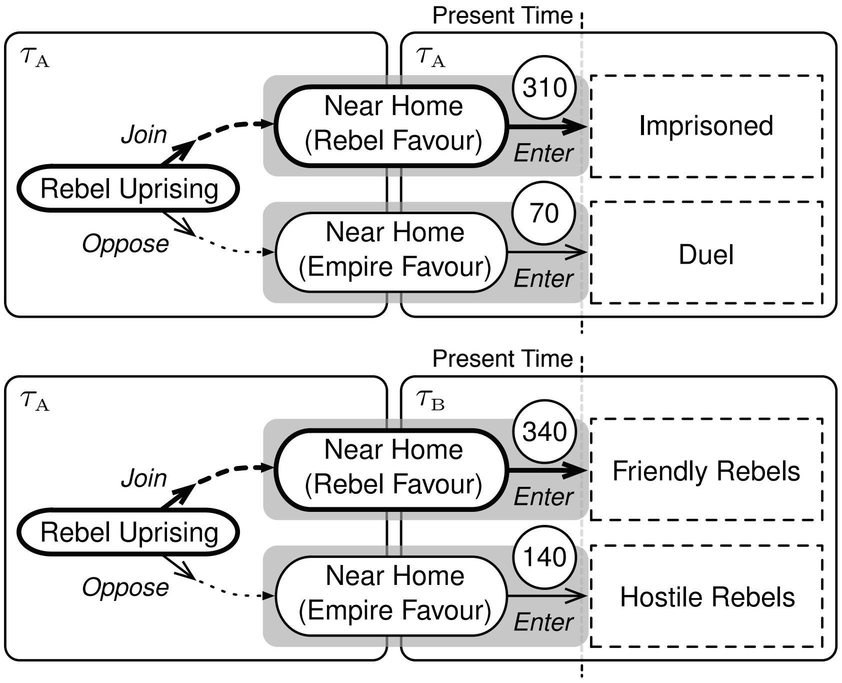


Figure 5.18: Part of *Lord of the Borderlands* represented as an MDP, showing the effects of choosing one of two available transition functions (τ_A or τ_B) at the highlighted outcomes. Top: The MDP if τ_A were selected. Bottom: The MDP if τ_B were selected. Bold paths show the current player's history. Circled values give each outcome's local desirability, given the indicated transition function. Dashed-edge boxes denote encounters whose details I have omitted to simplify the current example.

computes the local desirability of each outcome in the same way that PaSSAGE computes its objective function (i.e., using a player model and a conditional fun function; recall Table 5.4), I omit those parts of PaSSAGE 2's operation in the current example. Instead, I begin with assumed values for the local desirability of each outcome (one value for each available transition function – see the circled values in Figure 5.18) and demonstrate PaSSAGE 2's operation from there forward.

o	τ	Local ($des_L(o)$)	$\mathcal{A}_{o,\tau}$	$\alpha_\tau(o_E \mid o_R)$	Contextual ($des_C(o)$)	Obj. Func. (ϕ)
o_R	τ_A	310	$\{o_E\}$	0.80	192	**502**
	τ_B	340	$\{o_E\}$	0.30	60	400
o_E	τ_A	70				
	τ_B	140				

Table 5.6: Intermediate and final values for calculating the desirability of outcome $o_R = \langle$Near Home (Rebel Favour), *Enter*$\rangle$, assuming that the player reached o_R following a history h_t such that $\kappa(h_t) = \{\tau_A, \tau_B\}$. The highest overall desirability for o_R that can be obtained using any of the available transition functions is shown in bold. I use $w = 1$ in Equation 5.11. The additional parameters of des_L and des_C are omitted for brevity.

Referring to Figure 5.18, suppose that the player decided to join the rebels, resulting in Jaden arriving home with the favour of the rebels (outcome o_R). The third column of Table 5.6 shows how selecting either τ_A or τ_B following this outcome would affect its local desirability (des_L); 310 for

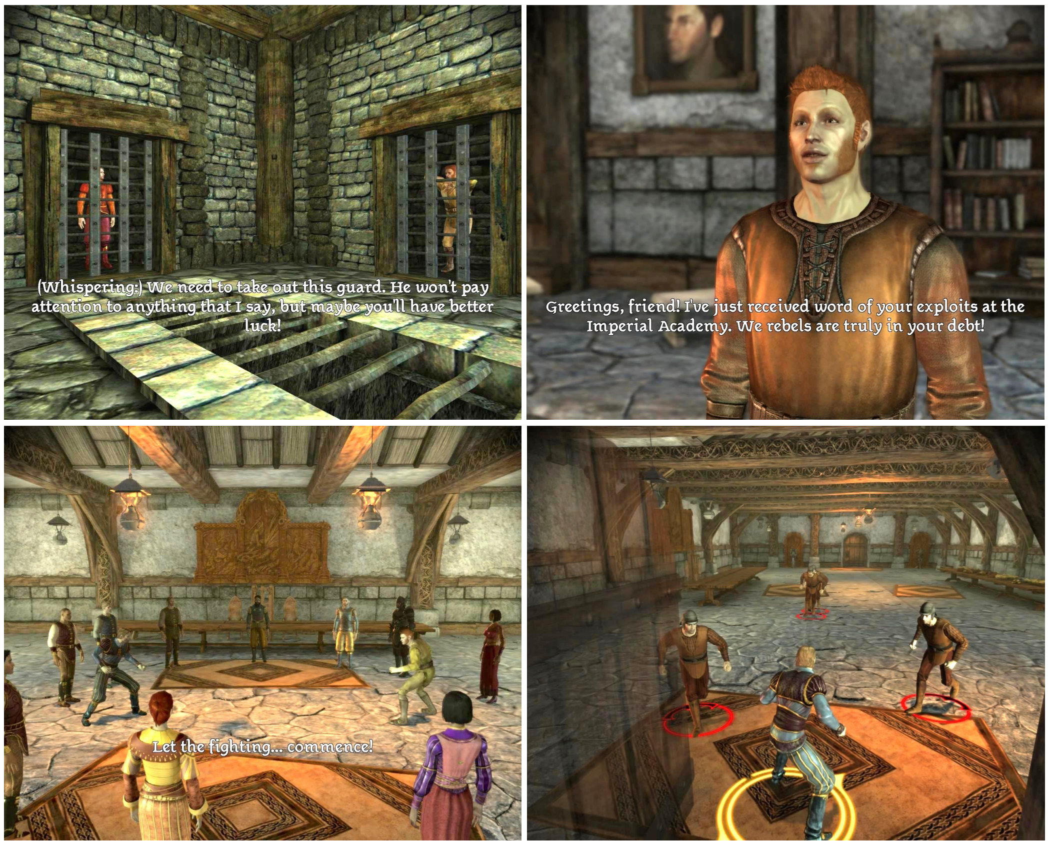

(Whispering:) We need to take out this guard. He won't pay attention to anything that I say, but maybe you'll have better luck!
Greetings, friend! I've just received word of your exploits at the Imperial Academy. We rebels are truly in your debt!
Let the fighting... commence!

Figure 5.19: Two encounters in *Lord of the Borderlands*. Top left: "Imprisoned". Bottom left: "Duel". Top right: "Friendly Rebels". Bottom right: "Hostile Rebels".

τ_{A}, and 340 for τ_{B}. Regardless of which transition function PaSSAGE 2 selects, the only alternative outcome to o_{R} is $o_{\mathrm{E}} = \langle$Near Home (Empire Favour), *Enter*$\rangle$, as shown in the fourth column ($\mathcal{A}_{o,\tau}$). To compute the contextual desirability of the current outcome ($\mathit{des}_{\mathrm{C}}(o_{\mathrm{R}})$), PaSSAGE 2 requires two more pairs of values. First, it needs the local desirability of o_{E} with respect to both τ_{A} (70) and τ_{B} (140), as shown in the bottom two rows of the table and the circles in Figure 5.18. Second, it needs the probability $\alpha_{\tau}(o_{\mathrm{E}}|o_{\mathrm{R}})$ that the events following o_{R} (given each transition function) would make the player aware of what *could* happen following o_{E} if the same transition functions were selected (fifth column: 0.80 for τ_{A}, and 0.30 for τ_{B}). Given all of the foregoing values, PaSSAGE 2 computes the contextual desirability of o with respect to both available transition functions (sixth column; Equation 5.10: e.g., $\mathit{des}_{\mathrm{C}}(o_{\mathrm{R}}, \tau_{\mathrm{A}}) = \frac{0.80}{1}(310 - 70) = 192$), and then it finally computes its objective function using Equation 5.11 (seventh column). Following these calculations, PaSSAGE 2 would select the transition function that maximizes its objective function for o_{R} (i.e., τ_{A}).

Comparing PaSSAGE 2 to PaSSAGE. This example highlights how PaSSAGE and PaSSAGE 2 operate differently to pursue different objectives: while PaSSAGE 2 would select τ_{A} to maximize *its* objective function (toward strengthening the player's belief that they have agency), PaSSAGE would instead select τ_B to maximize *its* objective function (toward maximizing the amount of fun that the player had). PaSSAGE would choose τ_B because PaSSAGE 2's computation of local desirability (des_L) is equivalent to PaSSAGE's objective function, and $340 > 310$ in column 3 of Table 5.6). The different behaviour between PaSSAGE and PaSSAGE 2 arises from the effect of contextual desirability (des_C) on PaSSAGE 2's objective function. Since choosing τ_A leads to both a fairly low local desirability for o_E (70) and a fairly high probability that the player would be aware of o_E (0.80), the contextual desirability computation for τ_A results in a fairly high value (192), modelling the player's relief at avoiding the less desirable o_E ($70 < 310$). By comparison, the contextual desirability computation for τ_B still models relief at having avoided o_E (since $140 < 340$), but the resulting value is lower because the estimate of the player's awareness of o_E is fairly low (0.30). When PaSSAGE 2 computes its objective function, τ_A's higher contextual desirability (versus τ_B) is enough to overcome its lower local desirability, which results in it being selected instead of τ_B.

Table 5.7 summarizes how PaSSAGE 2 can be represented using GEM.

PaSSAGE Component	**GEM Representation**
Player Actions	Player Actions
Probability of Player Actions	Estimated Player Policy (Block #4)
The Player's Experience	Histories
Gameplay Activities	Trajectories
Encounters	Groups of Trajectories
Player Model	Features (Block #5)
Gameplay Activity Annotations	Features (Block #5)
Bounded Rollouts	Rollout Function (Block #3)
Awareness Function	Features (Block #5)
Encounter Selection	Manager Policy, Decision Constraints (Block #1), & Objective Function (Block #2)

Table 5.7: A summary of how each of PaSSAGE 2's components can be represented using GEM.

Summary

In this chapter, I began by using GEM to represent two managers from traditionally disjoint sub-fields of AI: Weyhrauch's Moe from the sub-field of Drama Management, and *Left 4 Dead*'s AI Director from the sub-field of Dynamic Difficulty Adjustment. The fact that both of these managers can be fully represented in the GEM framework demonstrates its ability to unify existing work across these two sub-fields. I then continued to demonstrate GEM's versatility by using it to represent two managers that I created to manage story-based video games: PaSSAGE and PaSSAGE 2. Both PaSSAGE and PaSSAGE 2 can be implemented using all five GEM building blocks, with each manager using its own objective function to pursue its unique, intended effect (increasing fun for PaSSAGE, and strengthening beliefs in agency for PaSSAGE 2). Both managers use their decision constraint function in the same way (i.e., to retrieve a restricted set of transition functions that can be selected from), but the specific constraints used by each manager vary from one environment to the next. The rollout function and estimated player policy are shared across both managers. Rollouts are *bounded* in the sense that they only extend into the future up to the next decision point (or a game ending), and the player's policy is estimated to be such that the player will always perform the actions that seem to be the most fun to them at the time. The same feature vector is used across both managers, although the specific values that are returned depend on the environment being used and what trajectories are possible therein. Both managers use their feature vectors to automatically estimate a model of the player's preferences over different kinds of gameplay, toward maximizing the similarity between this model and the kinds of gameplay that could be used to continue the players experience. While PaSSAGE's objective function uses this similarity directly as an estimate of player fun, PaSSAGE 2's objective function uses it to model gameplay desirability, and then combines it with an estimate of the player's awareness of alternative sections of gameplay. It uses this combined information to model how players might feel regret or relief from having (respectively) missed a section of desirable gameplay or avoided a section of undesirable gameplay, and then estimates each player's feelings of agency as a linear combination of both models.

Chapter 6

Evaluation

In this chapter, I describe and demonstrate a general method for evaluating managers through controlled experimentation in human user studies. I say that this kind of evaluation is "gameplay-based" because it relies on having a group of players play through a given, managed game.

6.1 Gameplay-based Evaluation for Experience Managers

In this section, I discuss evaluation as the task of determining how well a given manager works – that is, how successful it is at affecting its players in the way that the designer intended. Answering this question involves three steps: (i) having a group of players experience trajectories in a given interactive environment, (ii) measuring and interpreting how successful the manager was at achieving its intended effect for trajectories that players played through, and (iii) analyzing the resulting data to determine how well one or more statistics of interest will generalize to a larger group of players (e.g., the amount of fun that players had, on average). In the subsections that follow, I discuss each of the foregoing steps and define several related concepts.

6.1.1 Groups of Players & Scores

Henceforth, a *score* is a measured value of how successfully a given player was affected by a trajectory that they experienced in a given game. Since gameplay is interactive, the player may have more influence than the manager has over the occurrence of any particular trajectory. As a result, it can be difficult to attribute any score that is measured for a trajectory to *only* the manager's operations. This problem can be alleviated by using the manager to manage the experiences of a group of players, measuring a score for each player, and then computing the average of the scores. I refer to this average as the *manager's score*. Ideally, any individual variations in player behaviour (which affects scores by influencing which trajectory occurs) will only have small influences on a manager's score in comparison to the manager's consistent behaviour across all players (assuming that its policy remains fixed).

Manager Scores & Sequences of Transition Functions

Every manager's score depends on the way in which it generates a *sequence of transition functions* (by selecting them at decision points during gameplay) given any stream of actions that a player performs. The order of the transition functions in this sequence is determined by the order in which they appear in the trajectory that describes the player's experience (Definition 6.1). Selecting sequences of transition functions is the only way for a manager to influence the scores that players give. Assuming that selecting different sequences for any player can result in them giving different scores, an *ideal manager* could maximize its score by (i) identifying the set of sequences for each player that would cause them to give the highest score and (ii) selecting transition functions according to any sequence in that set.

Definition 6.1 (Sequence of Transition Functions)**.** Given a trajectory $h \in H$, let q be a *sequence of transition functions* that describes, for each decision point in h, the transition function that immediately follows that point in h. The order of the sequence is given by the order in which the decision points appear in h.

Real versus Simulated Players

Previous evaluations of experience managers have used either real players (Barber and Kudenko 2007; Thue et al. 2007a; Sharma et al. 2007; Thue 2007; Ramirez and Bulitko 2014) or simulated players (Weyhrauch 1997; Magerko 2006; Nelson et al. 2006; Roberts et al. 2006) – that is, AI agents that perceive and act in the interactive environment in a way that (ideally) mimics the way that real players would behave in their place. Simulated players are more convenient to use than real players for logistical reasons (e.g., one need not schedule hundreds of real players to attend gameplay sessions), but the generalizability of the data that can be obtained from them relies heavily on how accurately they represent a real group of players. Using real players can simplify this aspect of generalization by drawing a sample of players from the experimenter's target group of players (i.e., the group to which the experimenter would like to generalize any results of the experiment). However, real players can also introduce noise into the gathered data from factors that are beyond the experimenter's control (e.g., someone might rate a manager as being very successful because they feel generous at the time, regardless of how the manager actually performed). Since obtaining and validating an accurate model of player behaviour for a novel game (to create reliable simulated players) would require gathering data from human players anyway (Yannakakis et al. 2006), the method for gameplay-based evaluation that I consider in this chapter uses human players directly (i.e., it does not create or use simulated players).

From the perspective of evaluating experience managers, a player p can be summarized as a tuple of two functions: a *score function* (σ_p), which gives their score for a given trajectory h, and a *policy* (π_p), which, given a history h, a state s, and an action a, gives the probability of the player performing action a in state s following history h. Definition 6.2 defines this tuple explicitly. The only way to compute these functions is have the player play the game and measure their score.

Definition 6.2 (Player)**.** A *player* is a tuple $p = \langle \sigma_p, \pi_p \rangle$, where $\sigma_p(h)$ gives the player's score for any given trajectory h (i.e., $\sigma_p : H \rightarrow \mathbb{R}$), and $\pi_p(h, s, a)$ gives the probability of the player performing action a in state s following history h (i.e., $\pi_p : H, S, A \rightarrow [0, 1] \in \mathbb{R}$).

6.1.2 Measuring Success & Complete Trajectories

The score that arises from a given player experiencing a given trajectory can be measured in several ways, including (i) gathering player self-reports (e.g., responses to survey questionnaires (Vermeulen et al. 2010)), (ii) directly observing the player (e.g., via biometric sensors (McQuiggan et al. 2006)), or (iii) indirectly observing the player through their in-game actions (e.g., making their avatar cheer after an in-game fight). Yannakakis and Togelius discussed the benefits and limitations of these methods in a survey paper (2011). In brief, they noted that player self-reports can provide rich information, but can suffer from experimental noise and can intrude on the player's experience in a way that disrupts the effect being measured (e.g., by pausing the game to administer a questionnaire). Direct observations can be less prone to player-generated noise, but may suffer from sensor-generated noise (e.g., inaccurate readings of skin conductivity) as well as *habitation* – the effect where a player's physiological response to the same stimuli diminishes with repeated exposure. Some sensors can also intrude on the player's experience by being inconvenient or unwieldy to use (e.g., electroencephalographic sensors). Indirectly observing the player (i.e., observing in-game actions) is the least intrusive method of measuring a trajectory's score, but doing so must assume that both (i) there exists a relationship between the player's in-game actions and how they were affected by the trajectory, and (ii) the experimenter can either access it directly or estimate it in some way.

Complete Trajectories

While gameplay-based evaluation does not depend on the method of measurement that is used, some methods require the experimenter to assume that every player's experience in that game will eventually end (e.g., post-game questionnaires). Formally, I define a *game ending* as a state from which any player action will only ever result in the same state occurring again (Definition 6.3). Whenever a trajectory h begins at time step $t = 0$ and contains exactly one ending of the game, I say that that trajectory is *complete* (Definition 6.4). Let $H_\mathrm{c} \subseteq H$ be the set of all complete trajectories that can occur in a given interactive environment e.

Definition 6.3 (Game Ending)**.** Given an interactive environment $e = \langle S, A, (\tau_0, \dots) \rangle$, a state s is said to be *ending of the game* iff the game can transition to no other state after s is reached (i.e., $\forall \tau \in \kappa(h_t), \forall a \in A, \forall s' \in S \backslash \{s\}$ [$\tau(s, a, s) = 1$ and $\tau(s, a, s') = 0$]).

Definition 6.4 (Complete Trajectory)**.** Given an interactive environment $e = \langle S, A, (\tau_0, \dots)\rangle$, a trajectory $h \in H$ through e is said to be *complete* iff h begins at time step $t = 0$ and contains exactly one game ending.

6.1.3 Interpreting Success

To sidestep the difficulty of interpreting a manager's score in absolute terms (recall Section 1.3.2), one can instead interpret it relative to the score of a different manager, toward ideally claiming positive progress in the development of new managers. Throughout this chapter, I refer to the former manager as a *target manager* and the latter as a *baseline manager*. For example, one might wish to determine whether a modified version of *Left 4 Dead*'s AI Director (the target) was better than the original version (the baseline) at manipulating the emotional intensity of its players.

Random Baseline Managers

I focus on the use of *random baseline managers* in this chapter. The general argument for doing so is as follows: if, given the same task, an AI system was shown to reliably outperform a system that operated randomly, then one could claim that the AI system demonstrates intelligence more convincingly than the random system does. In the context of experience management and gameplay-based evaluation, an experimenter can use a random baseline manager to test different hypotheses about a target manager's behaviour. Specifically, by selectively randomizing any of the target's building blocks (i.e., χ, κ, ϕ, ρ, $\tilde{\pi}$, or $\boldsymbol{f}$), a random baseline manager can be created that differs from the target manager in specific, experimenter-intended ways. For example, to test the hypothesis that a target manager's objective function (ϕ) contributes positively to its score, a random baseline manager could be created in two steps: (i) copy the target's policy and all building blocks *except* its objective function, and (ii) define a new objective function (for the random baseline to use) that gives pseudo-random values. Creating random baseline managers in this way helps to ensure the condition that the two competing systems being evaluated (AI-based vs. random, as above) are evaluated with respect to the same task. To determine whether or not one of them can reliably outperform the other, scores must be collected from players and compared via statistical analyses (Section 6.1.5).

6.1.4 Attributing Success: Player-Specific or Player-Independent Managers

Beyond the immediate effects of player actions, another concern with attributing success can arise when a particular kind of manager is evaluated in comparison to a random baseline manager; namely, a *player-specific* manager. Whenever the player's actions in a given game can influence which transition functions a given manager selects during gameplay, I say that the manager is *player-specific*. For example, all of the managers that I presented in Chapter 5 are player-specific, because the values given by either their decision constraint function or their objective function depend on the player's actions. If a manager is *not* player-specific, I say that it is *player-independent*. A player-independent manager can be thought of generally as a manager that samples randomly from a given probability distribution over transition functions at each decision point that occurs in the game. For example, the dynamic appearance of useful items in a First Person Shooter (e.g., medical kits or ammunition) could be driven by a player-independent manager that causes ammunition to appear 80% of the time and medical kits to appear 20% of the time (regardless of any player actions). If a player-independent manager's decision constraints are such that only one transition function is available to select following every state/action pair, I call that manager a *fixed manager*. Otherwise, I call it a *random manager*.

Every player-independent manager is either a fixed manager or a random manager, but not every random manager is player-independent. For example, the random baseline manager that I described in Section 6.1.3 might not be player-independent, because even though its objective function gives pseudo-random values, its decision constraint function (κ) might still restrict which transition functions can be selected in a way that depends on the player's actions.

For any manager to be player-independent, it must use a decision constraint function that is *balanced*. Intuitively, a decision constraint function is balanced if, regardless of the player's actions, (i) the same number of decision points occur for every player, (ii.a) all of the decision points that can be reached first during gameplay (i.e., before any other decision points) yield the same set of transition functions, and (ii.b) the same is true for every set of decision points that can be reached second, third, etc. thereafter. Definition 6.5 formalizes this concept.

Definition 6.5 (Balanced Decision Constraint Function)**.** A decision constraint function (κ) is said to be *balanced* iff:

1. every complete trajectory $h \in H_{\text{c}}$ contains the same number of decision points (i.e., $\forall h, h' \in H_{\text{c}} : \delta(h) = \delta(h')$), and
2. given any two complete trajectories $h, h' \in H_{\text{c}}$, the decision constraints at every ith decision point in both h and h' provide the same set of transition functions for the manager to choose from (i.e., $\forall h, h' \in H_{\text{c}} : \forall i \in [1, \delta(h)] : \kappa(h_i) = \kappa(h'_i)$).

where $\delta(h)$ gives the number of decision points in h (i.e., $\delta : H \rightarrow \mathbb{Z}^+$) and h_i denotes the prefix of h that ends at h's ith decision point.

Two Sources of Success for Player-Specific Managers

Given two player-independent managers, the only way that they could obtain different scores is by giving sequences of transition functions to players in different proportions. For example, if one of the managers gave sequences according to $\{q_{\text{A}} : 60\%, q_{\text{B}} : 40\%\}$ and the other gave them according to $\{q_{\text{A}} : 10\%, q_{\text{B}} : 90\%\}$, then which manager scored higher would depend on whether players who were given q_{A} generally gave higher or lower scores than players who were given q_{B}. Since every manager gives sequences to players in *some* proportions, some part of the success (or failure) of every *player-specific* manager can be attributed to those proportions. A player-specific manager might also score differently than another manger for a second reason: because it selects transition functions similarly to how an ideal manager would (recall Section 6.1.1), toward ensuring that each player receives a sequence of transition functions that yields the highest available score. Every player-specific manager thus has two sources of success: one from being player-specific, and one from giving generally high-scoring sequences to players more often than sequences that generally score poorly.

Proportional Random Variants

A particular kind of player-independent baseline manager can help the experimenter distinguish between a player-specific manager's two sources of success. Namely, by observing the proportions in which the player-specific manager gives sequences of transition functions to its players, a player-independent manager can be defined in such a way that it gives sequences in the proportions that were observed. I refer to such a manager as the *proportional random variant* of a given player-specific manager. For example, given a player-specific manager that gives one of two available sequences 60% of the time (e.g., $\{q_{\text{A}} : 60\%, q_{\text{B}} : 40\%\}$), a proportional random variant would randomly give sequences to players such that 60% of players received q_{A} and 40% of players received q_{B}. In general, when sequences are given to players in the same proportions across a target manager and a baseline manager, the effect of these proportions on the scores of both managers should be equal (within experimental noise). As a result, the only aspect of the two managers that *could* differentiate between their scores is the success that arises from their player-specific behaviour (if any). More concretely, by comparing a player-specific manager to its proportional random variant as a baseline, the experimenter can isolate the former manager's player-specific behaviour as the only only way that it could have influenced the difference between its score and that of the baseline. By doing so, the experimenter can answer the question: "Is the target manager's success affected by its player-specific behaviour?" The foregoing ideas about proportional random variants and their use in evaluating managers are novel contributions of this work.

Limitations. To preserve the utility of comparing a player-specific manager to its proportional random variant, the effect of giving any particular sequence of transition functions to a player must be the same across both managers. That is, the player's trajectory cannot depend on which manager gave them any particular sequence of transition functions. To ensure that this restriction is maintained, the player must not be able to identify which manager was used to manage their experience, and the proportional random variant's decision constraint function must itself satisfy certain restrictions. Specifically:

1. it must be balanced (recall Definition 6.5), and

2. it must define a set of decision points that is identical to the set of decision points that is

defined by the player-specific manager.

When combined, the foregoing restrictions place additional requirements on part of the player-specific manager. Since the same number of decision points must occur along every possible complete trajectory (because the proportional random variant's κ must be balanced; restriction 1) and both managers must share the same set of decision points (restriction 2), the player-specific manager must also ensure that every possible trajectory will include the same number of decision points. If this restriction were not enforced, it would be possible for the player-specific trajectory to give a sequence of transition functions to a player that could never be given by the proportional random manager (e.g., one that contained one too many transition functions), thereby preventing the managers from giving sequences to players in equal proportions. Thus, a proportional random variant can only be created for managers whose decision constraints ensure that the same number of decision points will occur in every complete trajectory.

6.1.5 Analysis & Generalization

Given a target manager, a baseline manager, a game to be managed, and a group of potential players, one can use the two managers to manage player experiences in the game, measure the scores that those players give as a result, and then compute each manager's score. To determine the extent to which any observed differences between the managers' scores can be generalized to a larger population of players, the gathered data can be used to estimate the expected value of each manager's score and obtain a degree of confidence that any difference between the two values exceeds a given amount (e.g., 0, to show a difference between the two expected values).

Expected Score

The *expected score* of a manager m is the expected value of the score σ that would arise from it managing the experience of any player in a given population P in interactive experience e (Equation 6.1):

$$\mathrm{EV}_P(\sigma|m,e) = \sum_{h\in H_c}\sum_{p\in P}\Pr(h,p|m,e)\sigma_p(h) \tag{6.1}$$

Using Bayes' Rule, Equation 6.1 becomes:

$$\mathrm{EV}_P(\sigma|m,e) = \sum_{h\in H_c}\sum_{p\in P}\Pr(h|p,m,e)\Pr(p|m,e)\sigma_p(h) \tag{6.2}$$

Computing $\mathrm{EV}_P(\sigma|m,e)$ using Equation 6.2 is not practically feasible, since one would need to measure $\sigma_p(h)$ for each complete trajectory in H_c, for every player in P. Even if the players in P varied their actions over multiple plays widely enough to experience every complete trajectory in H_c, the scores that they would give for all but their very first trajectory would be biased by their memories of the previous trajectories that they had experienced. Instead of measuring the score of every possible trajectory exhaustively, a sampling-based approximation to $\mathrm{EV}_P(\sigma|m,e)$ can be obtained instead. Specifically, by measuring the manager's score for only the *first complete trajectory* ($h_p^1 \in H_\mathrm{c}$) through e_t that is experienced by each player $p \in P$, each player can be used to draw a sample from the distribution of scores that would be obtained from an exhaustive measurement process. In terms of Equation 6.2, this sampling-based approximation can be represented mathematically by setting the first probability term ($\Pr(h|p, m, e)$) to one when $h = h_p^1$, and zero otherwise. Equation 6.3 shows the result of this operation, where $\widetilde{\mathrm{EV}}_P$ (i.e., EV_P with a $\sim$ symbol above) represents an *approximate expected score*.

$$\widetilde{\mathrm{EV}}_P(\sigma|m, e) = \sum_{p \in P} \Pr(p|m, e)\sigma_p(h_p^1) \tag{6.3}$$

Assuming that every player in P is equally likely to play through an experience in e managed by m, $\Pr(p|m, e) = \Pr(p) = 1/|P|$. With respect to players' first complete trajectories through a given environment e, a manager's expected score can thus be approximated by the arithmetic mean of the scores given by each player following their first complete trajectory h_p^1 through e:

$$\widetilde{\mathrm{EV}}_P(\sigma|m, e) = \sum_{p \in P} \frac{\sigma_p(h_p^1)}{|P|}. \tag{6.4}$$

Hypothesis Testing via Linear Regression

To obtain a confidence value for an observed difference between the approximate expected scores of two managers, the scores that players gave can be analyzed using linear regression. Although a detailed discussion of using linear regression for hypothesis testing is beyond the scope of this dissertation, the general idea is to measure how much one or more *independent variables* (e.g., $\mathbb{X}_1, \mathbb{X}_2, \ldots$) influence the value of a particular *dependent variable* of interest (e.g., $\mathbb{Y}$), under the assumption that the dependent variable is a linear function of the independent variables[1]. To test the hypothesis that a target manager's score differs from that of a baseline manager one would measure how much using one or the other (henceforth, the "manager variable", $\mathbb{M}$) influences the score that players provide ($\mathbb{S}$). To estimate the amount of influence that a given independent variable (e.g., $\mathbb{M}$) has over the dependent variable ($\mathbb{S}$), each variable in the analysis is mapped to one dimension of a multidimensional space, and a best-fit line through the data points is projected into the two-dimensional space given by $\mathbb{S}$ and $\mathbb{M}$; the slope of this projection then describes the difference between the variables (e.g., between the average values of each manager's score). Finding a positive slope would indicate that the target manager's score was higher (on average) than that of the baseline manager, at least for the players in the group that participated in the experiment.

In addition to estimating a slope $\mathcal{B}_\mathbb{X}$ for each independent variable $\mathbb{X}$, linear regression can also provide a real number in $[0, 1]$ (called a *p-value*) for each $\mathbb{X}$. In this context, a *p*-value is the probability of drawing a particular false conclusion: that the expected value of $\mathcal{B}_\mathbb{X}$ is at least as

[1]Additional assumptions are also required, descriptions of which are readily available both on the Internet and in statistics textbooks (e.g., see (Cohen et al. 2002)). I describe how I checked these assumptions in Section 6.2.5.

extreme as the slope that was estimated from the experiment's data, when in fact $\mathbb{X}$ has no linear relationship with the dependent variable $\mathbb{Y}$ (i.e., $\mathcal{B}_{\mathbb{X}} = 0$). To use linear regression to test the hypothesis that a particular $\mathbb{X}$ *does* have an effect on $\mathbb{Y}$ (e.g., that the expected value of the target manager's score will be higher than that of the baseline), one examines its associated slope and p-value. The lower the p-value, the more confidence one can have in the claim that there *is* a linear relationship between $\mathbb{X}$ and $\mathbb{Y}$, and thus that the scores of the two managers differ not only in the context of the experiment that was performed, but in more general contexts as well (e.g., all undergraduate students at a certain university). Seeing both a positive slope and a low p-value for the manager variable would offer support for the claim that the target manager yields higher scores; the lower the p-value, the stronger the support.

6.2 Case Study: Evaluating PaSSAGE & PaSSAGE 2

In the remainder of this chapter, I demonstrate the method of gameplay-based evaluation that I proposed in Section 6.1 by describing how I used it to evaluate both PaSSAGE (in *Annara's Tale*) and PaSSAGE 2 (in *Lord of the Borderlands*). Henceforth, whenever my discussion applies equally to both PaSSAGE and PaSSAGE 2, I will refer to them generally as "PaSSAGE", and make distinctions between them only when necessary.

At a high level, I evaluated PaSSAGE using the following procedure. First, I created an interactive environment and implemented the target manager using commercial video game technology. Second, I implemented a random baseline manager as a point of comparison. Third, I obtained and adapted survey instruments (i.e., questionnaires) that both collect demographic data (e.g., age, or gender) and measure the manager's success at achieving its intended effect (e.g., "Fun" on a scale of 1 to 5). Fourth, I recruited over one hundred participants to play through the target's testbed game, some with the target manager managing their experience, and some with the baseline manager managing their experience instead. Finally, I analyzed the resulting data using common statistical techniques, toward determining whether or not the target manager was more successful at achieving its intended effect than the baseline.

6.2.1 Research Questions, Hypotheses, & Experimental Designs

My evaluations of PaSSAGE explored two research questions: one that concerns the value of learning a model of each player, and one that concerns PaSSAGE's performance overall. After presenting each question, I state my hypothesis about its answer and then describe the experiment that I designed to test that hypothesis.

Research Question #1

> *Does assigning sequences of transition functions in a player-specific way contribute positively, negatively, or not at all to PaSSAGE's expected score?*

This question seeks to determine whether PaSSAGE being player-specific has a positive, neutral, or negative effect on its ability to assign high-scoring transition function sequences to its players. I hypothesized that PaSSAGE's player-specific operations contribute positively to its expected score (**Hypothesis 1**).

Experimental Design. Because Hypothesis 1 involves the player-specific aspects of PaSSAGE, it can be tested by comparing PaSSAGE to its proportional random variant. Before creating this baseline manager, however, I ensured that PaSSAGE's decision constraints caused every player trajectory to include the same number of decision points (and thereby met the restriction that I noted in Section 6.1.4).

Hypothesis 1 can be expressed as given in Equation 6.5, where ψ denotes PaSSAGE and β_ψ is its proportional random variant, and where the approximate expected scores being compared are defined as in Inequality 6.4:

$$\text{EV}_P(\sigma|\psi, e) > \text{EV}_P(\sigma|\beta_\psi, e). \tag{6.5}$$

Let Q be the set of all sequences of transition functions that could be selected by a given target manager (in my case, PaSSAGE) in a given interactive environment. To create β_ψ, I estimated the probabilities with which PaSSAGE assigns sequences of transition functions to players (i.e., $\Pr(q|\psi), \forall q \in Q)$) by randomly sampling a group of players from the population ($P_\psi \subset P$) and having PaSSAGE manage each player's experience. By counting the number of times that PaSSAGE assigned each sequence q to a player and dividing by the total number of players ($|P_\psi|$), I obtained an estimate of $\Pr(q|\psi)$ for every $q \in Q$. I then created β_ψ so that it would assign transition function sequences randomly according to the estimated distribution.

Experiment Overview. To estimate the expected values of ψ and β_ψ's scores, I drew two sample groups of players from the population (P_ψ and P_{β_ψ}) and had them play through a given interactive environment e. In all of my experiments, my population was a group of first year undergraduate Psychology majors at the University of Alberta who were required to participate in user studies for course credit (see Section 6.2.4 for more detail). As I described in the previous paragraph, players in P_ψ had their experience in e managed by PaSSAGE. Meanwhile, players in P_{β_ψ} had their experience managed by β_ψ. After playing, each player filled out a survey questionnaire, which I describe along with other details of my experimental procedure in Section 6.2.3.

To help avoid any bias that could have been caused by having all of the players in P_ψ complete their participation before any of the players in P_{β_ψ} (e.g., due to the first sessions occurring during the university's mid-term exam period), I created software to facilitate interleaving the participation of the two groups while still ensuring that $\Pr(q|\psi)$ and $\Pr(q|\beta_\psi)$ would remain equal for every sequence q (with slight variations due to occasional participant absences and technical faults). Specifically, for each player that was randomly selected to participate in P_ψ, my software recorded the sequence of transition functions that PaSSAGE assigned to them, and then assigned that sequence to the next player that was randomly selected to participate in P_{β_ψ}. As a result of this (automated) procedure, the distributions over the sequences that were assigned by PaSSAGE and by β_ψ remained approximately equal throughout each experiment.

Research Question #2

> *For a given population of players, is PaSSAGE's expected score higher, lower, or no different than the expected score of a uniform random manager?*

This is a question that a practitioner (e.g., a video game developer) might ask. Many commercial video games feature a large amount of content that players effectively discover at random while they explore the world (e.g., *Skyrim* (Bethesda Game Studios 2011) or *Grand Theft Auto V* (Rockstar North 2013)). Would using a manager to guide each player's experience result in them having more fun? I hypothesized that in the context of a story-based video game, PaSSAGE's expected score would be higher than that of a uniform random manager (**Hypothesis 2**). Mathematically, this hypothesis is given by Inequality 6.6, where β denotes a uniform random manager (see Definition 6.6):

$$\mathrm{EV}_P(\sigma|\psi) > \mathrm{EV}_P(\sigma|\beta). \tag{6.6}$$

Definition 6.6. Let a *uniform random manager* β be an experience manager that gives sequences of transition functions $q \in Q$ randomly, as drawn from a uniform distribution over Q.

Experimental Design. Testing Hypothesis 2 amounts to comparing the expected scores of ψ and β. Although these values could be compared by sampling players from the population as I explained for Hypothesis 1, the similarity between PaSSAGE's proportional random variant (β_ψ) and the uniform random manager (β) allows for a more efficient approach. Specifically, given a set of data that was collected from an experiment that tested Hypothesis 1 (i.e., scores for players in both P_ψ and P_{β_ψ}), the data that is required to test Hypothesis 2 can be automatically estimated using bootstrapping (Efron 1979), thereby avoiding the significant expense of collecting new data using a traditional experimental design. I explain my use of bootstrapping in the following section.

Bootstrapping Procedure. Bootstrapping is a data resampling technique that facilitates estimating population statistics (e.g., the expected population mean) from a set of data sampled from the population (Efron 1979). By repeatedly generating new samples (i.e., drawing data with replacement) from a given sample and calculating the desired statistic for each one, a confidence interval can be obtained that includes the true value of the statistic in the population, with a given probability. Importantly for this work, the resampling process can be *weighted*, toward ensuring that some cases in the original sample will be selected more frequently than others during each resampling step.

Managers β_ψ and β both assign transition function sequences to players in a random fashion, and they are both player-independent managers. The only difference between them is the proportions in which they assign each of the available sequences of transition functions to a group of players: β_ψ's proportions come from PaSSAGE, while β's proportions are all equal to one another $(1/|Q|)$. Therefore, weighted bootstrapping can be used to estimate the expected score of a uniform random manager, given a collection of scores that were measured for the proportional random variant (β_m) of any manager m. Although I used MATLAB's "bootci" function (The MathWorks Inc. 2013) to estimate both β's expected score and a confidence interval around it, I describe how weighted bootstrapping can be performed in Figure 6.1.

```
Input:  $[\boldsymbol{\sigma}, \boldsymbol{q}] = [\langle \sigma_i, q_i \rangle : 1 \le i \le |P_{\beta_m}|]$ : a matrix of the score given by each player
            $p_i \in P_{\beta_m}$ (column 1) and the sequence of
            transition functions that was assigned to
            them by $\beta_m$ (column 2)
        $n$ : the number of players in the sample (i.e., $|P_{\beta_m}|$)
        $Q$ : the set of all possible transition function sequences
        $\forall q \in Q, \Pr(q|\beta_m)$ : the probability of each sequence being given by $\beta_m$
        $b$ : the number of bootstrap replicates to generate
Output: $\widetilde{\mathrm{EV}}_P(\sigma|\beta, e)$ : the estimated expected score of a uniform random manager, $\beta$
1  $\boldsymbol{w} = \langle \left(|Q| \Pr(q_i|\beta_m)\right)^{-1} : 1 \le i \le n \rangle$
2  $\bar{\boldsymbol{\sigma}} = \langle \rangle$
3  for $i \in [1, b]$
4      $\boldsymbol{\sigma'} = \mathrm{sample}(\boldsymbol{\sigma}, n, \boldsymbol{w})$
5      $\bar{\sigma}_i = \frac{1}{n} \sum_{j=1}^{n} \sigma'_j$
6  end for
7  $\widetilde{\mathrm{EV}}_P(\sigma|\beta, e) = (1/b) \sum_{i=1}^{b} \bar{\sigma}_i$
```

Figure 6.1: Pseudocode for using weighted bootstrapping to estimate the expected score of a uniform random manager β from data collected from the proportional random variant β_m of some manager m, where bold variables represent vectors, σ_i represents the score that was measured for player p_i, and q_i represents the sequence of transition functions that β_m assigned to p_i.

In the pseudocode, I use $[\boldsymbol{\sigma}, \boldsymbol{q}] = [\langle \sigma_i, q_i \rangle : 1 \le i \le |P_\psi|]$ to represent the matrix of data that could be collected for β_m from a group of players $P_{\beta_m} \subseteq P$. On line 1, a *weight vector* ($\boldsymbol{w}$) is defined that will be used when generating new bootstrap samples; this vector describes a probability distribution over the rows in the data matrix. This probability distribution defines, for each data point, the probability of that data point being included in the sample that is generated on line 4. Intuitively, the more frequently any sequence q appears in the data matrix, the lower the weight of every data point that contains that sequence. Assigning weights according to line 1 ensures that the probability of including a data point with sequence q in the generated sample will be equal over all $q \in Q$. The probability of each sequence being given by the proportional random manager, β_m, can be estimated as I described in my experiment for answering Research Question #1. On line 2, a vector of average scores ($\bar{\boldsymbol{\sigma}}$) is defined to store the value of each *bootstrap replicate* – that is, the value of the statistic being estimated (in this case, the expected score) as calculated from the bootstrapped sample. Lines 3 to 6 perform the bootstrapping procedure b times. Each time, line 4 draws a random sample of size n from the vector of scores that were measured from β_m (i.e., $\boldsymbol{\sigma}$), according to the probability distribution defined by the weight vector ($\boldsymbol{w}$). The result is a bootstrapped sample of n scores ($\boldsymbol{\sigma'}$) that is then used in line 5 to calculate the bootstrap replicate for this iteration ($\bar{\sigma}_i$). On line 7, the expected score of the uniform random manager is then estimated as the mean of all b bootstrap replicates.

Using bootstrapping with weighted resampling according to the weights in $\boldsymbol{w}$ allowed me to estimate the data needed to test Hypothesis 2 from the data that I collected to test Hypothesis 1.

6.2.2 Testbed Implementation

Toward answering the questions that I posed in Section 6.2.1 through a set of controlled experiments, I implemented PaSSAGE and PaSSAGE 2 each in the context of a short, story-based video game (*Annara's Tale* for PaSSAGE, and *Lord of the Borderlands* for PaSSAGE 2)[2]. I created *Annara's Tale* using the Aurora Neverwinter Toolset (BioWare Corp. 2002) during my prior work (Thue 2007), and *Lord of the Borderlands* using the Dragon Age: Origins Toolset (BioWare Corp. 2011); BioWare Corp. created both tools as part of the development of commercial video games. I described *Annara's Tale* and *Lord of the Borderlands* in detail in Chapter 5, but as a brief review, both of them allowed the player to: (i) control a character in a virtual, 3-dimensional world, (ii) converse with computer-controlled, non-player characters (by choosing from lists of pre-authored responses), and (iii) interact with objects in the virtual environment (e.g., searching containers or repairing broken items). Every possible trajectory in *Annara's Tale* contains three opportunities for a manager to make decisions (i.e., three decision points), while every possible trajectory in *Lord of the Borderlands* contains two. In both games, every decision point offers the the manager two transition functions to choose from, meaning that $2^3 = 8$ transition function sequences can be assigned to players by managers of *Annara's Tale*, while $2^2 = 4$ sequences can be assigned by managers of *Lord of the Borderlands*. Both games take between 30 and 45 minutes to play.

6.2.3 Measuring Scores: Survey Instruments

Conducting the experiments that I described in Section 6.2.1 requires measuring the score that a manager achieves as a result of managing a given player's experience. To do so, I asked each player to complete a Likert-like survey (Likert 1932) after their experience was complete, on which they were asked to state their agreement with several statements on a discrete scale ranging from "Strongly Disagree" to "Strongly Agree". I used two such surveys during the course of this work: one that I used during my prior research (Thue 2007; Thue et al. 2007a; 2007b), and one that I adapted from more recent work by Vermeulen et al. (Vermeulen et al. 2010; Thue et al. 2011). Both surveys provided real-valued measures of a manager's score with respect to PaSSAGE's particular intended effects (i.e., how much fun the player had for PaSSAGE, and how strongly they believed

[2]*Annara's Tale* is an extension of my prior work (Thue et al. 2007a; Thue 2007), while *Lord of the Borderlands* is new.

that they had agency for PaSSAGE 2; recall Sections 5.3 and 5.4). Each survey also provided some demographic information about each player (e.g., age, gender, and familiarity with playing video games). I present these surveys in more detail in Appendix A.

6.2.4 Experimental Procedure

I conducted each of my experiments as follows. To begin, I recruited a group of participants from the Research Participation Pool that is administered by the Department of Psychology at the University of Alberta; these participants consisted of undergraduate university students who were enrolled in an introductory-level psychology course at the time, and they received partial course credit in exchange for their participation. I introduced all of the participants to the study both verbally (from a rehearsed script) and textually (from a prepared handout). Because having an AI experience manager guiding a game could be perceived to be novel (and thus bias the scores that players reported), I phrased the introductory material so that experience management was never mentioned; participants were instead given the impression that the study was simply investigating how different types of gameplay would affect their feelings of fun and agency in the context of a story-based video game. After giving their consent to participate in the experiment, every participant played through a brief tutorial that I created to help players become familiar with the controls for the game[3]. All of the participants were encouraged to ask for help if they encountered any problems. After successfully completing the tutorial, every participant played through a game (either *Annara's Tale*, or *Lord of the Borderlands*, depending on the experiment) with either PaSSAGE (ψ) or the proportional random variant (β_ψ) managing their experience. The manager that was used for each participant was determined randomly, thereby creating the player groups P_ψ and P_{β_ψ} that I needed to test my hypotheses (recall Section 6.2.1). After they finished playing, each participant completed a survey (Section 6.2.3) and received a debriefing that explained the true nature of the study (i.e., evaluating an AI experience manager) and why deceiving them was necessary[4]. The documents that I created for this study (including the introduction and consent forms, the survey forms, and the debriefings) can be found in Appendix A.

6.2.5 Data Analysis Techniques

The result of each experiment was a set of data that contained information about each player and the experience that they had in the game. For each player, this data contained:

- survey information consisting of:
  - the manager's score for that player (as reported by the player),

[3]Every participant had the option to complete a written assignment instead of playing the game, but none chose to do so.

[4]A small number of players were unable to complete the game during the hour that was allotted. These players completed the survey and were debriefed like any other player (to meet the guidelines of the Research Participation Pool), but I discarded their survey data to avoid any bias caused by having an incomplete experience.

  - their age,
  - their gender,
  - their general familiarity with playing video games (either "none at all", "less than one hour per week", "between 1 and 3 hours per week", "between 3 and 7 hours per week", and "more than 7 hours per week"),

- gameplay information consisting of:
  - the sequence of transition functions that the manager assigned them to, and
  - the values in PaSSAGE's player model at every time that it changed during the game.

Linear Regression

To test Hypothesis 1, I analyzed the data that I collected from each experiment using linear regression, as I described in Section 6.1.5.

Controlling for Gaming Familiarity. During my prior work on PaSSAGE, I found that players' general familiarity with playing video games can affect how much fun and agency they report having felt after playing through a given game (Thue et al. 2007b). To control for this effect when testing my hypotheses, I included an additional independent variable in my linear regression analysis, which I created from the self-reported familiarity with video games (henceforth "gaming familiarity", $\mathbb{G}$) that I collected on each player's survey. More specifically, I reduced the five categories that were available to select from on the survey down to only two: "unfamiliar" players were all those who reported that they played no video games at all in an average week, and "familiar" players were all those who reported that they spent at least some time playing video games in an average week. Pooling the data in this way was necessary because the general lack of players with gaming familiarity in the Research Participation Pool made it impractical to perform a more detailed analysis.

A Linear Model. The linear regression model that I used to test my hypotheses is shown in Equation 6.7, where $\mathcal{I}$ represents the intercept of the line, and the third variable ($\mathbb{V}$) models any interaction that might occur between manager and gaming familiarity (e.g., in my prior work, I found that PaSSAGE was more effective for players with low gaming familiarity than it was for highly familiar players (Thue et al. 2007b)). Recall that the independent variable $\mathbb{S}$ represents the score that each player provided after playing.

$$\mathbb{S} = \mathcal{I} + \mathcal{B}_{\mathbb{M}}\mathbb{M} + \mathcal{B}_{\mathbb{G}}\mathbb{G} + \mathcal{B}_{\mathbb{V}}\mathbb{V} \tag{6.7}$$

Since the data that I collected concerning which manager was used and how familiar each player was with playing video games was categorical in nature, I translated each category into a number for use in Equation 6.7 as given in Table 6.1. The variable $\mathbb{V}$ is computed as the product of the numbers that represent the values of $\mathbb{M}$ and $\mathbb{G}$. The specific values in the table ensure that the coefficients that are calculated for Equation 6.7 will represent the quantities that I wish to test (i.e., so that $\mathcal{B}_{\mathbb{M}}$ will represent the difference between the average values of PaSSAGE's and β_{ψ}'s scores) and also help to minimize co-linearity between $\mathbb{V}$ and the other independent variables (see Cohen's et al. (2002) work on "coding" for details).

Manager		**Gaming Familiarity**	
Category	$\mathbb{M}$	*Category*	$\mathbb{G}$
PaSSAGE	0.5	Familiar	0.5
Proportional Random	-0.5	Unfamiliar	-0.5

Table 6.1: Mapping from category to numerical value for two independent variables.

Bootstrapping

To test Hypothesis 2, I used bootstrapping to generate 8000 new samples[5] from the data that I originally collected for the proportional random variant in each of my experiments that tested Hypothesis 1. As I described in Section 6.2.1, I weighted the bootstrap selection probabilities of the original data points to ensure that every sequence of transition functions would have an equal probability of being represented by a data point in the new sample. Thus, each new sample simulated collecting data from a uniform random manager, as needed to test Hypothesis 2. For each sample, I used linear regression to compare its data to the data that I originally collected for PaSSAGE, thereby generating: (i) 8000 bootstrap replicates of the slope for the manager variable in Equation 6.7 ($\mathcal{B}_{\mathbb{M}}$), and (ii) a Studentized, $x\%$ confidence interval (Efron and Tibshirani 1993) around that slope as well. The confidence interval and the mean of each set of the bootstrap replicates allow Hypothesis 2 to be tested. The wider the confidence interval, the higher the confidence value, x, indicating that one can be more confident in claiming that the (true) value of $\mathcal{B}_{\mathbb{M}}$ is in that interval. Testing Hypothesis 2 amounts to finding the widest confidence interval around $\mathcal{B}_{\mathbb{M}}$ that does not include 0. The confidence value for this interval will indicate the degree of confidence that one can have in asserting that $\mathcal{B}_{\mathbb{M}}$ is non-zero. If the estimate of $\mathcal{B}_{\mathbb{M}}$ is positive, then Hypothesis 2 will be supported; the wider the interval, the stronger the support.

Data Cleaning

When conducting experiments that involve both human participants and complex computer software (such as commercial video game engines), a variety of problems can occur that can invalidate the data that was collected from one or more participants. For example, some players got their avatar permanently stuck on the roof of one of the houses in *Annara's Tale*, while others turned off their computer during the experiment. In other cases, players discovered that they could effectively skip past most of the in-game dialogue by rapidly clicking on their mouse. Doing so allowed them to

[5]I chose the number 8000 to balance computation time against the accuracy of the results; the results varied by approximately 0.005% in successive runs.

finish their participation more quickly, but it left them with very little knowledge about what had happened in the game, and that in turn may have had a detrimental effect on their enjoyment or how strongly they felt their agency. To identify such problematic data points, I noted any problems that occurred while the experiment was underway (through direct observation and automated log analysis), and then examined each player's gameplay logs after they had finished playing to verify that a problem had indeed occurred. I "cleaned" the data by omitting all verified, problematic data points from my analyses.

Outlier Detection

For the purposes of this work, an *outlier* is one of a set of data points (each point representing the data that was collected from one participant) that is abnormal with respect to the true relationship (if any) that exists between the variables in a linear regression. Identifying outliers is an important part of preparing to analyze data from an experiment that examines experience managers, because the personal circumstances of each participant can influence the scores that they report, and analyzing data that contains unusually extreme scores (in comparison to other players) can lead to erroneous inferences about the managers being studied. Given the occasionally low motivation and attention levels among participants from the Research Participation Pool (as demonstrated by, for example, their checking cell phones while playing, or rushing to finish the game as quickly as possible), I decided to remove any detected outliers from the data before performing the regression analysis.

I used Tukey's outlier labelling rule to identify cases that were extreme with respect to the independent variable in my experiment (score) (Tukey 1977). Compared to other methods of identifying extreme points in data, Tukey's method has the advantage of having a relatively high *breakdown point* (Huber 2004) of approximately 25%, that is, at least 25% of the data set must be outliers before the method will risk identifying arbitrary cases as outliers or non-outliers. In contrast, methods that are based on the mean and standard deviation have a breakdown point of 0% (i.e., a single outlier can cause arbitrary identifications).

For a given set of data, I applied Tukey's rule four times – once for each pairing of the possible values of the variables for manager and gaming familiarity (i.e., $\langle$PaSSAGE, unfamiliar$\rangle$, $\langle$PaSSAGE, familiar$\rangle$, $\langle\beta_\psi$, unfamiliar$\rangle$, and $\langle\beta_\psi$, familiar$\rangle$). Each of these pairings represents a particular subset of the participants in the experiment (e.g., everyone unfamiliar with gaming who played a game managed by PaSSAGE), and applying Tukey's rule to them independently assumes that each of these subsets is a random sample of an equivalent subset of the population. I removed any identified outliers from the data set before proceeding with my analyses; in the experimental results that follow (Section 6.3), I describe any data points that were removed from each experiment.

Assumption Checking

After removing any detected outliers from the data, I checked for violations of the assumptions of linear regression using the methods recommended by Cohen et al. (2002). Briefly, these methods include: (i) visually examining scatter plots of each independent variable against the regression residuals[6] to verify that the relationship between the dependent and independent variables is approximately linear, (ii) using Levene's test to verify that the variance of the residuals remains constant across the values of each variable in the experiment (e.g., from unfamiliar to familiar for gaming familiarity), and (iii) visually examining normal q-q plots to verify that the residuals follow a normal distribution. In Section 6.3, I describe the results of these tests for each experiment that I conducted.

6.3 Experiments & Results

In this section, I continue my case study of PaSSAGE and PaSSAGE 2 by presenting the results of four experiments that I ran to test my hypotheses about them. More specifically, I performed two experiments for each manager: one using the survey instrument that I used in my prior research (Thue 2007), and one using an instrument that I adapted from Vermeulen et al.'s work (2010) (recall Section 6.2.3). Henceforth, I will refer to these surveys as "survey 1" and "survey 2", respectively. Since these surveys measure scores along different ranges of values (a 7-point Likert-like scale for survey 1 versus a 5-point Likert-like scale for survey 2), I present all of the scores in this section using a normalized, real-valued scale from zero (lowest score) to ten (highest score).

To simplify comparing my results across Hypotheses 1 and 2, I express how confident one can be in an experiment's support for its hypothesis as a *confidence value* between 0% and 100%. For experiments that test Hypothesis 1, the confidence value is equal to 1 less the *p*-value that is given by the linear regression that I performed. For experiments that test Hypothesis 2, I report the confidence value of the widest confidence interval around the estimated slope of the manager variable ($\mathcal{B}_{\mathbb{M}}$) that did not include zero (because bootstrapping yields confidence intervals instead of *p*-values).

PaSSAGE				PaSSAGE 2			
Survey 1		Survey 2		Survey 1		Survey 2	
H1	H2	H1	H2	H1	H2	H1	H2

Table 6.2: A map showing the organization of my experiments in this chapter. H1 and H2 denote sections that discuss my tests of Hypotheses 1 and 2, respectively. For PaSSAGE, see Section 6.3.1. For PaSSAGE 2, see Section 6.3.2.

Table 6.2 shows the organization of the following sections, in order from left to right. I begin by presenting my results for PaSSAGE, and follow with my results for PaSSAGE 2. For each manager,

[6]For each player in an experiment, a *residual* is computed as the difference between the value of the dependent variable that was observed for that player and the value that is predicted by the linear model (i.e., by evaluating the right hand side of Equation 6.7).

I first consider data collected using survey 1, and then consider data collected using survey 2. For each data set, I report the results of testing Hypotheses 1 and 2.

Holm-Bonferonni Adjustment

To avoid the biased p-values and resulting errors of inference that can arise from performing multiple statistical tests on a single set of data, I have adjusted the confidence values that I present in the following sections using the Holm-Bonferonni procedure (Holm 1979). Specifically, since my tests of Hypotheses 1 and 2 both use the same set of data for each pairing of manager and survey (e.g., the data from PaSSAGE, Survey 1), a correction for having performed two tests is required for each pairing. The Holm-Bonferonni adjustment for two tests proceeds as follows: A p-value is obtained (or can be computed[7]) from each of the two tests; let x be the lowest of the two p-values, and let y be the other p-value. The adjusted p-value for x is then $2x$, and the adjusted p-value for y is $\max(y, 2x)$. Each adjusted confidence level is then equal to 1 less the corresponding adjusted p-value.

6.3.1 PaSSAGE

In the context of evaluating PaSSAGE (ψ), Hypothesis 1 can be stated as follows: PaSSAGE's player-specific operations contribute positively to PaSSAGE's expected score as measured in terms of self-reported player fun. Similarly, Hypothesis 2 can be stated as follows: PaSSAGE's expected score will be higher than that of a uniform random manager, in terms of self-reported player fun.

Results for Survey 1

Using survey 1, I conducted an experiment following the experimental design that I presented in Section 6.2.1. Specifically, 186 participants from the Research Participation Pool played through *Annara's Tale*. Half of the players had their experience managed by PaSSAGE, and half had it managed by the proportional random variant for PaSSAGE. For each participant, I obtained a data point as described in Section 6.2.5. After cleaning the data as I described in Section 6.2.5, data points from 133 participants remained: 75 for PaSSAGE, and 58 for the proportional random variant. The average age of the participants was 19.4 years; 40 were male, 90 were female, and 3 chose to not specify their gender. Tukey's outlier labelling rule identified no outliers, and the assumptions of linear regression were met.

Hypothesis 1. Table 6.3 shows the results of testing Hypothesis 1 using the linear regression analysis that I described in Section 6.2.5, with the Fun score from survey 1 serving as the dependent variable.

The first three columns show the number of players that were present in each of the subgroups of the experiment. The fourth column gives the mean score for each manager (out of 10), adjusted

[7]Given a confidence interval obtained from bootstrapping, its corresponding p-value can be computed as: p-value $=$ $1 -$ confidence.

Manager	Gaming Familiarity (Number of Players)		Estimated Marginal Mean	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	p-value	Adjusted Confidence
	Familiar	Unfam.				
ψ	36	39	5.23	0.52	0.221	77.9%
β_ψ	33	25	4.71			

Table 6.3: Results for testing Hypothesis 1 with respect to PaSSAGE (ψ), using survey 1's measure of Fun as the score being evaluated; β_ψ is PaSSAGE's proportional random variant.

for the fact that the numbers of high versus low familiarity players in the sample data were not perfectly balanced[8]. The fifth column gives the estimated slope for the manager variable in the linear model from Equation 6.7 ($\mathcal{B}_{\mathbb{M}}$), and the sixth column reports a two-tailed p-value for testing Hypothesis 1. Since the estimated slope is positive (indicating a higher score for PaSSAGE than the proportional random variant), the results of this experiment support Hypothesis 1 with 77.9% confidence ($1 - 0.221$).

Hypothesis 2. To test Hypothesis 2, I generated 8000 random samples of the data that I collected in my experiment that tested Hypothesis 1, with sampling weights set such that each sample would approximate a set of data collected from a uniform random manager (β). Table 6.4 shows the results of testing Hypothesis 2 using this bootstrapping analysis (see Section 6.2.5 for details), with the Fun score from survey 1 serving as the dependent variable.

Manager	Number of Players	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	Confidence Interval	Confidence	Adjusted Confidence
ψ	75	0.77	$[0.01, 1.04]$	97%	94%
β	58				

Table 6.4: Results for testing Hypothesis 2 with respect to PaSSAGE (ψ), using survey 1's measure of Fun as the score being evaluated; β is a uniform random manager.

In the second column of the table, the value 58 does not indicate that 58 players had their experience managed by the uniform random manager; in this and all later tables that presents results from testing Hypothesis 2, the number of players that is listed for the uniform random manager actually describes the number of players whose experiences were managed by the proportional random manager (β_ψ), and whose data were used as the input to the bootstrapping process that I described in Sections 6.2.1 and 6.2.5. The third column gives the estimated slope for the manager variable as in Table 6.3. The fourth and fifth columns gives the range and adjusted confidence level of the widest confidence interval around the estimated slope that does not include zero. The given level thus states my confidence in stating that the true slope is different from zero, and thus that PaSSAGE's expected score is different from that of a uniform random manager. To reduce computation time, I restricted my search for this interval to integer-valued confidence level.

[8]This adjustment occurs during linear regression because gaming familiarity is a variable in the linear model ($\mathbb{G}$), and the results are the estimated marginal means in column 4 of Table 6.3.

Since the estimated slope is positive and a 94% confidence interval does not include zero, the results of this experiment support Hypothesis 2 with 94% confidence.

Results for Survey 2

Similarly to how I conducted my experiment that tested Hypothesis 1 using survey 1, I recruited another group of players from the Research Participation Pool (104 in total) and had them play through *Annara's Tale*. After cleaning the data, 77 data points remained: 33 for PaSSAGE, and 44 for the proportional random variant. Tukey's outlier labelling rule indicated one outlying data point in the proportional random variant / familiar subgroup (score: 0.23 out of 10). The average age of the participants was 19.2 years; 18 were male, and 58 were female. The assumptions of linear regression were met.

Hypothesis 1. Table 6.5 shows the results of testing Hypothesis 1. Since the estimated slope is negative, the results of this experiment do not support Hypothesis 1. On the contrary, these results provide some evidence for *refuting* Hypothesis 1 with 90.2% confidence ($1 - 0.098$); I will discuss this result along with the others in Chapter 7.

Manager	Gaming Familiarity (Number of Players)		Estimated Marginal Mean	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	p-value	Adjusted Confidence
	Familiar	Unfam.				
ψ	16	17	4.40	-0.90	0.098	90.2%
β_{ψ}	13	30	5.30			

Table 6.5: Results for testing Hypothesis 1 with respect to PaSSAGE (ψ), using survey 2's measure of Fun as the score being evaluated; β_{ψ} is PaSSAGE's proportional random variant.

Hypothesis 2. Table 6.6 shows the results of testing Hypothesis 2 with respect to the Fun score from survey 2. Since the estimated slope is negative, the results of this experiment do not support Hypothesis 2, and instead refute it with 96% confidence.

Manager	Number of Players	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	Confidence Interval	Confidence	Adjusted Confidence
ψ	33	-0.78	$[-1.46, -0.03]$	98%	96%
β	43				

Table 6.6: Results for testing Hypothesis 2 with respect to PaSSAGE (ψ), using survey 2's measure of Fun as the score being evaluated; β is a uniform random manager.

6.3.2 PaSSAGE 2

In the context of evaluating PaSSAGE 2 (ψ_2), Hypothesis 1 can be stated as follows: PaSSAGE 2's player-specific operations contribute positively to PaSSAGE 2's expected score as measured in terms of self-reported player perceptions of *agency*[9]. Similarly, Hypothesis 2 can be stated as follows: PaSSAGE 2's expected score will be higher than that of a uniform random manager, in terms of self-reported player perceptions of agency.

Results for Survey 1

I recruited 97 participants from the Research Participation Pool and had them play through *Lord of the Borderlands*; PaSSAGE 2 managed the experience for approximately half of the players, while the remaining players' experiences were managed by the proportional random variant. After cleaning the data, 94 data points remained: 50 for PaSSAGE 2, and 44 for the proportional random variant. Tukey's outlier labelling rule identified three outlying data points in the proportional random variant / familiar subgroup (all with score 3.33 out of 10). The average age of the participants described by the remaining 91 data points was 21 years; 63 were female and 28 were male. The assumptions of linear regression were met.

Hypothesis 1. Table 6.7 shows the results of testing Hypothesis 1 with respect to the Agency score from survey 1. Since the estimated slope is negative, the results of this experiment do not support Hypothesis 1, and instead refute it with 89.3% confidence $(1 - 0.107)$.

Manager	Gaming Familiarity (Number of Players)		Estimated Marginal Mean	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	p-value	Adjusted Confidence
	Familiar	Unfam.				
ψ_2	30	20	6.82	-0.82	0.107	89.3%
β_{ψ_2}	19	22	7.64			

Table 6.7: Results for testing Hypothesis 1 with respect to PaSSAGE 2 (ψ_2), using survey 1's measure of Agency as the score being evaluated; β_{ψ_2} is PaSSAGE 2's proportional random variant.

Hypothesis 2. Table 6.8 shows the results of testing Hypothesis 2 with respect to the Agency score of survey 1, where PaSSAGE 2 is compared to a Uniform Random manager. Since the estimated slope is negative, the results of this experiment do not support Hypothesis 2, and instead refute it with 98% confidence.

Manager	Number of Players	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	Confidence Interval	Confidence	Adjusted Confidence
ψ_2	50	-0.76	$[-1.61, -0.001]$	99%	98%
β	41				

Table 6.8: Results for testing Hypothesis 2 with respect to PaSSAGE 2 (ψ_2), using survey 1's measure of Agency as the score being evaluated; β is a uniform random manager.

[9]Recall that PaSSAGE 2 targeted agency, while PaSSAGE targeted enjoyment ("fun").

Results for Survey 2

To test my hypotheses for PaSSAGE 2 using survey 2, I had 342 participants from the Research Participation Pool play through *Lord of the Borderlands* – PaSSAGE 2 managed the experience of approximately half of the players, while its proportional random variant managed the experiences of the rest. After cleaning the data, 323 data points remained: 172 for PaSSAGE 2, and 151 for the proportional random variant. Tukey's outlier labelling rule identified 12 outlying points: four in the PaSSAGE 2 / Familiar subgroup (with scores 0, 0.42, 1.25, and 2.08 out of 10, respectively), three in the PaSSAGE 2 / Unfamiliar subgroup (scores: 0.83, 1.67, and 2.08), two in the Proportional Random / Familiar subgroup (scores: 1.67 and 2.92), and three in the Proportional Random / Unfamiliar subgroup (scores: 0.83, 1.25, and 1.25). After these outliers were removed, data points from 323 participants remained (mean age: 19.1 years; 95 male & 228 female), and the assumptions of linear regression were met.

Hypothesis 1. Table 6.9 shows the results of testing Hypothesis 1 for PaSSAGE 2 using the Agency score from survey 2. Since the estimated slope is positive, the results of this experiment support Hypothesis 1 with 60% confidence (after adjustment, given the result of testing Hypothesis 2).

Manager	Gaming Familiarity (Number of Players)		Estimated Marginal Mean	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	p-value	Adjusted Confidence
	Familiar	Unfam.				
ψ_2	93	71	7.52	0.22	0.262	60%
β_{ψ_2}	82	65	7.30			

Table 6.9: Results for testing Hypothesis 1 with respect to PaSSAGE 2 (ψ_2), using survey 2's measure of Agency as the score being evaluated; β_{ψ_2} is PaSSAGE 2's proportional random variant.

Hypothesis 2. Table 6.6 shows the results of testing Hypothesis 2 with respect to the Agency score from survey 2, based on 8000 weighted bootstrapping samples of the data that I originally collected for testing Hypothesis 1. Since the estimated slope is positive, the results of this experiment support Hypothesis 2 with 60% confidence.

Manager	Number of Players	Estimated Slope ($\mathcal{B}_{\mathbb{M}}$)	Confidence Interval	Confidence	Adjusted Confidence
ψ_2	164	0.20	$[0.01, 0.39]$	80%	60%
β	147				

Table 6.10: Results for testing Hypothesis 2 with respect to PaSSAGE 2 (ψ_2), using survey 2's measure of Agency as the score being evaluated; β is a uniform random manager.

6.3.3 Summary of Results

Table 6.11 summarizes the results of the experiments that I ran to test my hypotheses.

Manager	Survey	Total Number of Players	Hypothesis	Measured Score	Adjusted Confidence
PaSSAGE (ψ)	1	133	1	Fun	77.9%
			2	Fun	94%
	2	76	1	Fun	Ref. (90.2%)
			2	Fun	Ref. (98%)
PaSSAGE 2 (ψ_2)	1	91	1	Agency	Ref. (89.3%)
			2	Agency	Ref. (96%)
	2	311	1	Agency	60%
			2	Agency	60%

Table 6.11: A summary of my results for testing Hypotheses 1 and 2 with respect to PaSSAGE (measuring Fun) and PaSSAGE 2 (measuring Agency). Confidence values indicate that the corresponding hypothesis was supported or refuted at the given level (possible refutations are indicated by "Ref.").

The first column shows which manager I evaluated, compared to its proportional random variant for Hypothesis 1, and a uniform random manager for Hypothesis 2. The second column shows which survey instrument I used, and the third column shows the total number of players that participated in each experiment. The fourth, fifth, and sixth columns indicate whether or not the results of each experiment support the given hypothesis; if so, the given confidence value states how confident one can be in asserting that the hypothesis is true with respect to the measured score (or false, as indicated by "Ref." for "refutation").

6.3.4 Discussion of Results

The empirical results for my evaluations of PaSSAGE and PaSSAGE 2 were mixed. When the amount of fun that players had was measured using survey 1, PaSSAGE succeeded at increasing player fun. However, when fun was measured using survey 2, PaSSAGE failed to increase player fun and actually decreased it instead. Similarly, when the strength of players' beliefs in their own agency was measured using survey 2, PaSSAGE 2 succeeded at strengthening those beliefs. However, when this strength was measured using survey 1, PaSSAGE 2 failed to increase it and decreased it instead.

Potential Sources of Bias

In the following sections, I discuss several possible explanations for the mixed nature of the results that I obtained. In each section, I support or deny the stated explanation with either arguments or empirical evidence. To simplify my writing, I will refer to each of my four experiments using the abbreviations shown in Table 6.12.

Differences between Surveys. As can be seen in Appendix A, the two surveys that I used during my experiments differ in how they measure both how much fun players had and how strongly they believed that they had agency while they played. One possible explanation for the apparent conflict

Abbreviation	Experiment	
	Manager	Survey
A1	PaSSAGE	1
A2	PaSSAGE	2
B1	PaSSAGE 2	1
B2	PaSSAGE 2	2

Table 6.12: Abbreviations for each of my four experiments. In the first column, "A" denotes PaSSAGE, "B" denotes PaSSAGE 2, "1" denotes survey 1, and "2" denotes survey 2.

among my results is that the two surveys actually measure different concepts from one another (i.e., perhaps the kind of fun that survey 1 measures is different from the kind of fun that survey 2 measures, and likewise for agency). If this hypothesis were true, the results would no longer be in conflict with one another[10]. One observation that lends support to this hypothesis is that survey 1 asked for ratings of enjoyment relative to "an average video game of similar length that [the player had] played in the past" (or the player's expectation of one), while survey 2 asked for ratings of enjoyment that could be interpreted in a more absolute sense (e.g., requiring the player to agree or disagree with statements such as "my story experience was gratifying", without explicitly prompting them to compare their experience with any other).

To investigate whether surveys 1 and 2 measure different concepts from one another, I had 182 of the players in my sample group for experiment B2 complete *both* surveys while measuring both fun and agency on each survey. As a result, I obtained 182 four-tuples of data that had the following form: $\langle$Fun score from survey 1, Agency score from survey 1, Fun score from survey 2, Agency score from survey 2$\rangle$. Using this data, I computed both continuous (Pearson) and ranked (Spearman) correlation coefficients between two pairs of score distributions: the Fun scores from survey 1 versus the Fun scores from survey 2, and the Agency scores from survey 1 versus the Agency scores from survey 2. The resulting coefficients are given in Table 6.13.

Measured Score	Pearson Correlation	Spearman Correlation
Fun	0.60	0.56
Agency	0.70	0.70

Table 6.13: Correlation coefficients for Fun and Agency scores across surveys 1 and 2. For each coefficient, its associated p-value was less than 0.01.

Using terminology from Cohen (1988), the correlation between the surveys' scores for both Fun and Agency was strong to very strong (> 0.5). The strength and positive sign of these correlations (along with their p-values < 0.01, as computed using SPSS (IBM Corp. 2013)) provides evidence that the scores from both surveys represented similar interpretations of both fun and agency, and thus that the two surveys did not differ substantially in this regard.

The foregoing evidence should be considered preliminary, since the order in which all players

[10]One could still not interpret the results without first understanding the differences between the different concepts.

completed the surveys was the same: all players completed survey 2 first and survey 1 thereafter[11]. Thus, players' responses to survey 1 may have been biased by having just responded to survey 2. A future experiment could evaluate the similarity between the surveys' interpretations of fun and agency by having one group of players (group X) complete survey 1 before survey 2, and another group of players (group Y) complete survey 2 before survey 1. Doing so would result in four data sets: X1, X2, Y1, and Y2 (e.g., X1 indicates the data obtained from group X via survey 1). Finding strong correlations between all pairs of these data sets (i.e., X1 versus Y1, X1 versus X2, etc.) would provide stronger evidence that the surveys capture similar notions of fun and agency.

Differences in Player Group Composition. Although differences in players' prior familiarity with gaming was controlled for by the statistical model that I used for my analyses (i.e., in Equation 6.7), other differences between the player groups for each experiment could have affected the measured scores. More specifically, the greater the difference between the compositions of the player groups across two experiments (e.g., from A1 to A2), the higher the likelihood that the measured scores could differ for reasons other than the planned differences between the experimental and control groups (e.g., comparing PaSSAGE to its proportional random manager). For example, if the group of players obtained for A1 contained few males but the group obtained for A2 contained mostly males, then a gender-specific effect of PaSSAGE (e.g., "PaSSAGE increases the amount of fun that men report, but decreases the amount of fun that women report") could bias the computation of the slopes and confidence values during linear regression.

Using the demographic data that was recorded on the surveys and the gameplay logs from each experiment, I verified that the distributions of the player groups obtained for A1 and A2 did not differ significantly with respect to player gender (equal within 10%)[12], nor with respect to the sequences of transition functions that the managers assigned (equal within 10%). Comparing the same distributions of player groups from B1 and B2 yielded similar results (genders equal within 2%; sequences equal within 3%). The similarity of these distributions across the two surveys (1 and 2) for both managers (A: PaSSAGE, and B: PaSSAGE 2) suggests that the results of my experiments were not biased by differing distributions of either player gender or the sequences of transition functions that occurred during gameplay.

To test whether some other difference between the player groups in each experiment (beyond gender or sequence distributions) might have biased the results of my experiments, I performed a meta-analysis of each manager's scores as follows. First, I linearly mapped the scores for fun and agency that were recorded on survey 1's scale (range: 1 to 7) to fit survey 2's scale (range: 1 to 5) using the following function: $(f(x) = (x - 1) * 4/6) + 1$. The appropriateness of using a linear function for this mapping depends on two assumptions: 1) that the two surveys measure the same

[11]My priority was to obtain as much unbiased data as possible for testing PaSSAGE 2 using survey 2, and switching the order in which the surveys were presented (to have players complete survey 1 first) could have biased the scores that I collected for survey 2.

[12]For example, the proportion of males was 30.1% in A1, while in A2 it was 23.7%, and $30.1\% - 23.7\% < 10\%$. I computed these values using the CrossTabulations feature in SPSS (IBM Corp. 2013).

notions of fun and agency, and 2) that the minimum and maximum values of fun and agency on one survey are analogous to the minimum and maximum values on the other survey. I discussed the first assumption in Section 6.3.4, but whether or not the second assumption holds remains an open question. Next, I combined the data points that I obtained using survey 1 (with remapped scores) with the data points that I obtained using survey 2 to create a pooled data set (e.g., I combined the data points from A1 and A2). I then expanded my statistical model (from Equation 6.7) to include the experiment (e.g., A1 versus A2) as an independent variable, $\mathbb{E}$. Equation 6.8 shows this model, where $\mathbb{M} \times \mathbb{E}$ is a variable that represents an interaction between manager and experiment. Finally, I performed a linear regression analysis (following the procedure that I described in Chapter 6) to estimate slopes and *p*-values for this new model, using the pooled data from each version of PaSSAGE and its proportional random manager.

$$\mathbb{S} = \mathcal{I} + \mathcal{B}_{\mathbb{M}}\mathbb{M} + \mathcal{B}_{\mathbb{G}}\mathbb{G} + \mathcal{B}_{\mathbb{V}}\mathbb{V} + \mathcal{B}_{\mathbb{E}}\mathbb{E} + \mathcal{B}_{\mathbb{M}\times\mathbb{E}}(\mathbb{M} \times \mathbb{E}) \tag{6.8}$$

When comparing the two experiments that I ran using PaSSAGE (i.e., A1 and A2), the *p*-value for $\mathcal{B}_{\mathbb{M}\times\mathbb{E}}$ was 0.084. When comparing the two experiments that I ran using PaSSAGE 2 (i.e., B1 and B2), the *p*-value for $\mathcal{B}_{\mathbb{M}\times\mathbb{E}}$ was 0.018. These low *p*-values provide strong evidence that the results that I obtained were biased by at least one, as-yet-unknown difference between the player groups across the pair of experiments that I conducted for each manager. Furthermore, the fact that the *p*-values were low for $\mathcal{B}_{\mathbb{M}\times\mathbb{E}}$ (and not $\mathcal{B}_{\mathbb{E}}$; $p > 0.25$) shows that the combined effect of these unknown differences between groups substantially alters how experiences managed by PaSSAGE and PaSSAGE 2 are scored in comparison to those selected by their proportional random managers.

Extensions & Limitations. It would be very valuable to identify the source(s) of bias that I described in Section 6.3.4. Doing so with high confidence will require either good experimenter intuitions or a very large amount of data, as I describe in the following two paragraphs.

Given good intuitions, an experimenter could selectively test each of a few potential sources of bias with roughly similar numbers of participants as I had in each of my two meta-analyses (209 for A1 vs. A2, and 402 for B1 vs. B2). For PaSSAGE 2, a potential source of bias is how much players in each experimental group were motivated to feel control (i.e., agency) at the time that they participated in my experiments. Specifically, Burger (1986) found empirical evidence that people who have a higher desire for control over a given situation are more likely to overestimate the amount of control that they actually have. If the distributions (across experimental groups) of players' desire for control differed for experiments B1 and B2, then my results could have been biased in a way that produced the values that I observed.

Alternatively, a wide range of demographic and situational data could be gathered from every player in an experiment, and each element of data could be represented as a variable in an expanded statistical model. Each of the new variables could then be tested for interactions between the manager variable from my analyses. Unfortunately, such an approach would require a very large number of players to identify any potential source of bias. For example, with only ten new binary variables being investigated, data from at least 4096 players would be required to have at least one data point for each of the 2^{12} combinations of manager, gaming familiarity, and the ten new elements of data; this number would increase with the dimensionality of the data elements being tested. Although this scale of data collection may be out of reach for *Annara's Tale* and *Lord of the Borderlands* (due to practical limitations on how they can be deployed), it might be feasible for published interactive environments (such as commercial video games) that capture statistics from thousands or more players every week. For example, according to publicly available statistics from the digital distribution service Steam (Valve Corporation 2014), *Left 4 Dead 2* (the sequel to *Left 4 Dead*, which I described in Section 5.2) had more than 5000 *simultaneous* players on July 18, 2014. It would be very interesting to evaluate the AI Director of *Left 4 Dead 2* using controlled experiments similar to the ones that I described in Section 6.2.1, toward better understanding how player reports of fun and agency can be biased by different external factors.

General Findings & Limitations

The mixed nature of the results of my experiments limits the extent to which they can be generalized beyond the groups of players that participated in each study. Specifically, the current data cannot support any general claim that PaSSAGE or PaSSAGE 2 will yield higher scores than their proportional random managers or uniform random managers. At best, the results show that these managers have *some* effect in comparison to the baselines that I compared them to (with confidences values as stated below), but whether or not these effects are generally beneficial or detrimental remains an open question. Overall, my data supports the following claims with regard to the general population of undergraduate students at the University of Alberta:

- With at least 77.9% confidence (Table 6.11, row 1), PaSSAGE has an effect on player reports of fun that differs from that of its proportional random manager.
- With at least 94% confidence (Table 6.11, row 2), PaSSAGE has an effect on player reports of fun that differs from that of a uniform random manager.
- With at least 60% confidence (Table 6.11, row 7), PaSSAGE 2 has an effect on player reports of agency that differs from that of its proportional random manager.
- With at least 60% confidence (Table 6.11, row 8), PaSSAGE 2 has an effect on player reports of agency that differs from that of a uniform random manager.

As I argued in Section 6.3.4, the direction of the effects that PaSSAGE and PaSSAGE 2 have on their respective scores (either positive or negative) depends on at least one external factor that remains to be identified. As a result of this dependence, support for Hypotheses 1 and 2 remains unclear for both managers.

Summary

In this chapter, I presented a general, gameplay-based method for evaluating experience managers, wherein players play through a given game in two groups. In one group, a *target manager* manages each player's experience, and in another group, a *baseline manager* is used instead. A *score* is measured for each player (capturing how successful each manager was at achieving the target manager's intended effect), and the scores for each manager are averaged to calculate that *manager's score*. By applying statistical techniques to the gathered data, the probability of observing similar manager scores from larger groups of players can be assessed, toward generalizing the evaluation's findings to a wider population. I distinguished between managers that are *player-specific* and those that are *player-independent*, and noted that player-specific managers might obtain higher (or lower) scores than a player-independent manager for two reasons: because they give sequences of transition functions to players in a player-specific way, or because they give generally high-scoring sequences to players more often than those that score poorly. I introduced the *proportional random manager* as a baseline that can be used to distinguish between these two effects, and demonstrated its use across four user studies spanning hundreds of human players (two of PaSSAGE, and two of PaSSAGE 2). Although the results of these studies were inconclusive, I described how future gameplay-based evaluations might help to clarify our understanding of both PaSSAGE and PaSSAGE 2.

Chapter 7

Discussion

In this chapter, I discuss the benefits and limitations of Generalized Experience Management (GEM) and offer ideas for future work. I begin by considering GEM overall, and then follow with some specific lessons that I have learned from implementing and evaluating PaSSAGE and PaSSAGE 2.

7.1 Generalized Experience Management

To the best of my knowledge, GEM is the first mathematically defined framework for experience management that spans both drama management and dynamic difficulty adjustment. As such, it supports the first thesis statement of this work (recall Section 1.5). As I demonstrated in Chapter 5, the building blocks of GEM provide a common set of mathematical terms with which such managers can be represented. This is an important step toward better understanding the similarities and differences between managers, and ultimately integrating the diverse techniques that are developed by researchers across the field. Recent work by Ramirez and Bulitko (2014) supports this claim, for GEM helped them to identify how the building blocks of two existing managers could be usefully combined: the decision constraints that are generated by Riedl & Stern's (2006) Automated Story Director (Block #1), and the feature vector and objective function that PaSSAGE uses to maximize player fun (Blocks #2 and #5). Their resulting manager, PAST, showed promising performance in user study evaluations (Ramirez et al. 2013; Ramirez and Bulitko 2014). Following this work, Poo Hernandez and Bulitko (2014) developed a new manager, PACE, that extended PAST's feature vector with an appraisal-based model of player emotions, and aimed to influence the player's emotions in a designer-specified way.

7.1.1 Benefits & Implications for Research and Practice

The core idea of GEM is that for any given game, a manager's influence on the experiences that players have in that game can be represented as changing the game's transition function. This representation is flexible in several important ways:

- First, since changing a game's transition function subsumes changing that game's state, GEM allows for a unified representation of both drama managers (which often focus on changing the game's state) and systems that dynamically adjust the game's difficulty (which often focus on changing the game's dynamics).

- Second, GEM does not depend on the details of how a game's transition function is implemented, and so allows a variety of different implementation techniques (e.g., behaviour trees or programmatic scripts) to be thought of and represented in a unified way.
- Third, since GEM is grounded in the theory of MDPs, GEM managers have the opportunity to benefit from the body of existing research that studies MDPs (see Section 7.1.4).
- Fourth, because the designer can control precisely which transition functions can be selected by the manager at any point in the game (via decision constraints), they can retain as much control over each player's experience as they desire. I suspect that this opportunity for control will be particularly important for bridging experience management research to more industry-focused projects, since it allows the designer to rely on a manager's objective function as lightly or as heavily as they see fit.
- Fifth, even if the results of a GEM manager's policy cannot be obtained at some point during the game (e.g., due to a lack of computing resources or a communication failure), the player's experience can still continue uninterrupted using the transition function from the previous time step. Such a constraint on real-time performance is likely to be important in many commercial video games, where pausing to wait for the manager's computation would work counter to the game's intended effect (e.g., player excitement, or fun).
- Sixth, changing a game's transition function (instead of only its subsequent state) can simplify the creation of a manager's policy (as I discussed in Section 2.1.4), and using a feature vector (GEM Block #5) to handle player histories in aggregate can simplify this process even further.

For the foregoing reasons, I expect GEM to be a useful tool for academics and industry professionals, both for exploring new ways to manage player experiences and for putting research into practice in improving real-world interactive environments.

7.1.2 Mathematical Affordances

As a mathematical framework, GEM offers several affordances beyond its ability to generalize across managers from different domains. First, it is both *precise* and *concise*, allowing a manager's operation to be clearly and compactly specified using mathematical notation (as I demonstrated in Chapter 5). Second, GEM offers the ability to *analyze* and *predict* the behaviour of new managers; by examining the building blocks of existing managers and imagining different combinations thereof, the potential benefits of new managers can be estimated prior to their construction. This process led Ramirez et al. (2012) to combine of blocks from PaSSAGE (Thue et al. 2007a) and the

Automated Story Director (Riedl and Stern 2006), toward mixing PaSSAGE's pursuit of player fun with the re-planning capabilities of ASD. Finally, given a collection of building blocks that have been described in GEM's function-based notation (i.e., $\kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}$), GEM describes how a manager policy that uses the given blocks can be *derived* from the simple manager policy that serves as GEM's foundation (i.e., from Equation 7.1). Specifically, the desired policy can be obtained by evaluating the given functions in the context of Equation 7.2 for each state/action pair that is observed during the game (as I described in Section 4.2). Taken together, these affordances have the potential to simplify and accelerate future collaborations between groups that study and build new managers.

$$\tau_{t+1} = \chi(e_t, h_t, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{7.1}$$

$$\chi(e_t, h_t, \kappa, \phi, \rho, \tilde{\pi}, \boldsymbol{f}) = \operatorname*{arg\,max}_{\tau \in \kappa(h_t, \rho, \tilde{\pi}, \boldsymbol{f})} \phi(\tau, h_t, e_t, \rho, \tilde{\pi}, \boldsymbol{f}) \tag{7.2}$$

7.1.3 Assumptions & Limitations

For a given interactive environment, GEM relies on four key assumptions: (i) that the manager's success at affecting players as the designer intended depends on which history each player experiences, (ii), that there is only one player, (iii) that the manager can observe the states and actions that occur in the environment (i.e., the player's history) and, (iv) that the potential increase in how successfully the environment affects its players is large enough[1] to make management worthwhile.

History-dependent Effects

When using GEM, the designer must ensure that the intended effect of an interactive environment can actually be achieved by experiencing histories therein. Put differently, the set of trajectories that can occur in a GEM-managed environment must be effective for at least *some* potential players. GEM assumes that such content is given (as it would be in any traditionally-created environment), and supports the creation of managers that can shape each player's experiences of the given content while the game is underway. As I stated in Chapter 1, several other ongoing research efforts are exploring the challenge of procedural content generation (see Yannakakis & Togelius's (2011) survey), and I view their work as being complementary to GEM.

Single Player Experiences

I assumed that each managed game has only a single player to simplify my presentation of GEM, but extending it to multiplayer settings is conceptually straightforward, requiring only a redefinition of trajectories to use joint player actions (e.g., as composed from every player's individual action at each time step)[2]. *Practically*, however, implementing a GEM manager that effectively manages

[1]Both the magnitude of the increase and the final level of success are important, since even a large increase from "terrible" to "bad" might still make management seem not worthwhile.

[2]*Left 4 Dead* supports multiple players without this step, since it only considers the states in any given trajectory.

multiplayer games could be difficult, since a sequence of transition functions that successfully affects one player might be unsuccessful at affecting another. Finding the best way to balance this optimization process over multiple players simultaneously is a candidate for future work. Video game company CCP Games has expressed interest in pursuing this research in the context of their massively-multiplayer game *Eve Online* (CCP Games 2003).

Observable Player Histories

This assumption should be straightforward to meet in virtual interactive environments (e.g., video games or training simulators), provided that the software that implements the game's transition function is capable of recognizing when the (potentially abstract) states and actions of a player's history occur and then communicating that information to the manager. In *Annara's Tale* and *Lord of the Borderlands*, I accomplished this recognition with a set of simple scripts that would execute whenever certain events were captured by the game engine in which each game was built (the Aurora and Eclipse engines, respectively). For example, the state "Alone in Forest" was recognized in *Annara's Tale* by a script that executed when the player entered part of the game's forest.

For interactive environments that are situated in the real world (e.g., assisted living scenarios, augmented reality games, or amusement parks (Mehta et al. 2010)), the observability of the player's history will depend on both the availability of sufficiently reliable and accurate sensors, and the efficacy of the software that recognizes states and actions of the game given this raw sensor data. For example, the guitar-teaching software *Rocksmith 2014* (Ubisoft 2013) depends on custom hardware (a "RealTone cable") to convert an analogue audio signal from a standard guitar into a digital signal that is suitable for identifying the pitch and volume of every musical note that the student plays. Bonardi attributed the success of *Rocksmith 2014* as an interactive teaching tool to its ability to accurately recognize the player's actions on any standard guitar (Bonardi 2014).

Sufficient Opportunities for Success

In some interactive environments, it might *impossible* for a GEM manager to score higher than a given baseline manager, simply because the effects of the histories that are possible in that environment are too similar to one another for experience management to yield any difference in the player's score. For example, in the extreme case of no variation among the effects of the environment's histories, there would be no value in switching from any baseline approach to any GEM manager. Furthermore, if a given baseline approach was particularly successful, then the amount of possible improvement could be too small to be worth the effort of implementing a GEM manager. It might be possible to measure the success of a baseline approach through evaluations similar to those that I described in Chapter 6. If multiple histories could be evaluated by each player (with corrections for bias from previous plays), then the variance of the scores over histories could give some indication of how much improvement could be gained by switching to a GEM-based approach.

For example, if most players tended to give scores that varied between 4 and 8 (on a 10-point scale) and a baseline approach obtained an average score of 6, then a GEM manager could aim to provide players with the histories that they chose to score at 8 (instead of histories that score lower).

7.1.4 Future Work

When devising a framework that is meant to be general (like GEM), it is important to balance the tradeoff that exists between the framework's representational scope and its representational depth. A wide scope (i.e., encompassing many managers) can be achieved easily with a trivially shallow depth (e.g., simply stating that "every manager is an agent"), but a framework with such shallow depth would be of little use as an aid in analyzing or creating different managers. Similarly, great depth (where many fine details are captured) can be obtained by restricting a framework's scope to a single manager, but such a framework could be very difficult to apply when analyzing any other manager. Three ways in which GEM could be extended relate to (i) assessing the width of its scope, (ii) increasing the depth to which it can represent each manager, and (iii) exploring how it might inform and leverage existing research into Markov Decision Processes.

Assessing Representational Power

I have used GEM to represent four managers (Weyhrauch's Moe, the AI Director from *Left 4 Dead*, PaSSAGE, and PaSSAGE 2), but whether or not it can be used to represent other managers remains an open question. GEM's ability to represent Moe suggests that managers that arose from follow-up research may be representable as well, including *Façade*'s drama manager (Mateas and Stern 2003), Declarative Optimization-based Drama Management (DODM) (2006), and Targeted-Trajectory-Distribution MDPs (TTD-MDPs) (Roberts et al. 2006). Similarly, GEM's ability to represent *Left 4 Dead*'s AI Director suggests that the related directors in *Left 4 Dead 2* and *Alien Swarm* may be representable as well. Finally, GEM's ability to represent PaSSAGE suggests that PaSSAGE 2 (as I have already demonstrated), PAST (Ramirez et al. 2013; Ramirez and Bulitko 2014), and PACE (Poo Hernandez et al. 2014) might also be representable. Because of their similarity to the managers that I have already represented, I suspect that the foregoing managers constitute "low hanging fruit" in terms of being easily representable using GEM. Other potential candidates to investigate from academia include Young et al.'s Mimesis (2004), Magerko's Interactive Drama Architecture (IDA) (2006), Barber & Kudenko's Generator of Adaptive, Dillema-based Interactive Narratives (GADIN) (2007), Riedl et al.'s Automated Story Director (ASD) (2008), Swartjes et al.'s Virtual Storyteller (2009), Yu & Riedl's Prefix-Based Collaborative Filtering system (PBCF) (2012), and the managers in Arinbjarnar & Kudenko's Directed Emergent Drama (DED) (2008) and Weallans et al.'s Distributed Drama Management (DDM) (2012). Potential candidates from the video games industry include the managers from *SiN Episodes: Emergence* (Ritual Entertainment 2006; Kazemi 2008) and *Darkspore* (Maxis Software 2011).

Looking beyond domains that are related to games, it would be valuable to investigate whether GEM could be used to represent intelligent tutoring systems in the domain of education (Nwana 1990) or recommender systems in the domain of marketing (Graves et al. 1995). Existing research that explores experience management in the context of education (Rowe et al. 2010) and that applies collaborative-filtering techniques to drama management (Yu and Riedl 2012) offer promise for this avenue of future work.

Greater Representational Depth

One way in which the depth of GEM's representation of managers could be extended relates to the differences among the transition functions that managers consider during gameplay. In the current version of GEM, managers can compare transition functions by examining the trajectories that they might cause the player to experience. When two transition functions are largely identical, this examination can be made more efficient with higher-level knowledge about the difference between the two functions. A categorization of differences between transition functions similar to the "moves" of Weyhrauch's Moe could provide this knowledge (recall Section 5.1), while simultaneously allowing each GEM manager's operations to be represented in more detail. For example, Moe's future hints (Figure 7.1) make only a very localized change to the game's transition function while leaving the rest of the transition function unchanged (e.g., redirecting action 1 from state B to state E).

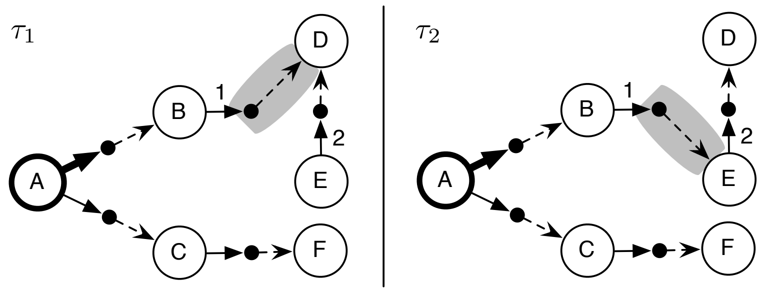


Figure 7.1: Representing a Moe *future hint* as changing a transition function (repeated from Figure 5.4 for convenience). When the transition function is changed from τ_1 to τ_2, the player will be forced to observe a hint in state E after performing action 1. While the transition probabilities following state/action pair $\langle$B, 1$\rangle$ are different, the rest of τ_1 and τ_2 are identical.

As I presented GEM in Chapter 4, this change would be represented by switching between two complete (and mostly identical) transition functions. Of all of the ways that the game's transition function could change after the first function is selected, it seems likely that these changes would be fairly minor, since the current transition function was (hopefully) chosen well by the manager at some earlier point in the player's experience (e.g., based on decision constraints or an objective function). By formally representing the differences between transition functions as first-class entities in GEM, one could potential build a library of *transition modifiers* as a generalization of Moe's moves, and further simply comparing different managers.

It might also be possible to combine GEM with frameworks that extend beyond the task of experience management, such as Tychsen's framework for multiplayer, pen-and-paper role-playing games (Tychsen 2008).

Integrating MDP Research

GEM's grounding in the theory of MDPs enables the application of existing research to benefit experience management, and doing so remains as future work. One direction that could be promising is to leverage Ng and Russell's (2000) or Syed et al.'s (2008) work on automatically learning a user's reward function and policy from observations of their actions in the context of MDPs. Having access to models of a player's reward function and policy could allow new interactive environments and managers to be evaluated (in a preliminary sense) based on only a symbolic representation of their states, actions, and transition functions, by using the player's policy to generate actions and the reward function to measure a score for each simulated player. This ability to explore the success of a manager without fully implementing a testbed could be very valuable in both industry and academia, allowing new ideas to be tested and improved at a faster pace than human subject testing allows. In particular, such a method might help designers manage the tradeoff that exists between reaching a wider audience and having to create more game content to do so, by identifying the smallest set of transition functions that would be needed to reach a given range of players.

7.2 Evaluating Experience Managers

Throughout the course of this work, I learned that evaluating experience managers can be challenging for several reasons. Some of these reasons are general to all kinds of evaluations, while others are specific to evaluations that seek to determine how successful a manager was at achieving the designer's intended effect.

7.2.1 Implementation Challenges

Implementing an interactive environment to use as a testbed can require an enormous amount of work, both in learning new technological tools and developing enough game content to support a manager's operation. For example, the single PaSSAGE 2 decision that I demonstrated in Section 5.4.4 required ten diverse segments of gameplay: one for each of the player's alternatives of opposing or joining the rebels (2 total), and one for each of the different ways that the player could react to each of the four potential outcomes of that decision (4 outcomes with 2 reactions per outcome). Each of these segments involved roughly 10 to 15 minutes of exploring 3D areas, collecting items, and interacting with non-player characters. In total, *Lord of the Borderlands* had 27 such segments, and took approximately 13 person months to develop (two undergraduate students – Trevon Romanuik and Charles Crittenden – helped me with this work). Even with this amount of effort, each player's experience only included two decision points (i.e., two opportunities to choose a transition function); expanding to allow more decision points would have required even more segments of gameplay. Riedl et al. (2008) and Ramirez & Bulitko (2014) mitigated the problem of content creation by reducing the complexity of each segment of gameplay (e.g., Ramirez & Bulitko used only text descriptions of story scenes and player actions), but such simplifications (particularly text descriptions) may only be appropriate for managers that focus more heavily on storytelling and narrative, with no need for the graphical and real-time components of typical video games. Since the player model used by PaSSAGE and PaSSAGE 2 relies heavily on the kinds of gameplay that are common in role-playing games, it was important to include similar kinds of gameplay in both *Annara's Tale* and *Lord of the Borderlands*.

7.2.2 Logistical Challenges

It can be difficult to acquire a sufficiently large sample of a particular population of interest without introducing bias. The Participant Research Pool at the University of Alberta partially mitigates this problem through a combination of single-blind, advance questionnaires (i.e., participants are unaware of which experiment is associated with each question) and double-blind assignments of participants to experiments based on the questionnaire data (e.g., participants who liked baseball were preferentially assigned to experiments run by Lee et al. (2014), toward testing their automated sports commentary system). Unfortunately, the use of advance questionnaires can severely limit the number of participants that are available (e.g., perhaps only 30% of participants in the pool enjoy watching baseball). Furthermore, the typical demographics of this group (roughly 19 years old and roughly two thirds female, as observed in my experiments) may have limited generalizability to other demographics of interest (e.g., children aged 10 to 15). Obtaining participants from other demographics requires recruiting from outside of the participant pool, which typically introduces self-selection bias by allowing participants to choose whether or not they will participate based on some initial description of the experiment. While self-selection may be desirable in some cases (e.g., a video game company might want testers who are already excited about their game), it is less desirable in others (e.g., when trying to assess whether a manager can improve the experiences of atypical players).

7.2.3 Challenges to Measuring Intended Effects

Once a group of players has been obtained to evaluate a given manager, it may be difficult to determine how successfully a *particular aspect* of that manager's operation affects its players (e.g., PaSSAGE's player-specific adaptations). When constructing a baseline manager to perform this kind of measurement, it is important to carefully consider what it means for the targeted aspect of the manager to be "switched off", and how doing so will affect each player's experience. The method of gameplay-based evaluation that I presented in Chapter 6 offers guidance with respect to how baselines managers can be defined and constructed, which I demonstrated in my evaluations of PaSSAGE and PaSSAGE 2. Poo Hernandez et al. (2014) intend to use this method to evaluate their manager (PACE) in a forthcoming study.

Given a manager to evaluate and a baseline manager to compare it to, reliably measuring how successfully each manager affected its players as the designer intended can be very challenging. As I learned from my evaluations of PaSSAGE and PaSSAGE 2, external factors can have substantial effects on the results obtained from survey questionnaires, and I suspect that identifying these factors will require extensive investigations. As a result of this influence, the second thesis statement of this work remains unsupported (Section 1.5); determining whether GEM can be used to build successful managers remains as future work. Furthermore, it is not clear that a single measurement captured once at the end of the player's experience (as I did in my evaluations) is the best way to evaluate a player's experience as a whole. For example, players may forget the earlier parts of a particularly long experience by the time it finally ends, even though those earlier parts were particularly effective (e.g., fun for the player) at the time. On the other hand, a player's likelihood of returning to a game after their first experience (which could be very important for commercial applications) could depend heavily on how they (would have) evaluated it after the end of that experience. Future work can explore these and other aspects of measurement and player behaviour.

Chapter 8

Conclusion

This work has been a multidisciplinary investigation of AI *experience managers* – computational systems that monitor and modify an interactive environment toward affecting a player therein in some designer-intended way. Focusing on the domain of computer video games, it has involved the creation, analysis, and evaluation of a variety of different managers, including one that co-founded its field of study, one from a successful commercial game, and two that I designed and implemented in original, story-based adventures. Along the way, I have combined formalism from computer science (the GEM framework), the creative work of game design (as testbeds for given managers), and the tools of applied statistics (for controlled evaluation), toward better understanding how managers can perform a task that is more often analyzed from a psychologist's point of view.

As a result of this investigation, I have made several contributions to the field. First, I presented Generalized Experience Management, the first mathematical framework for experience management that is simultaneously general enough to represent a variety of different managers (as I demonstrated in Chapter 5), and specific enough to suggest how the building blocks of existing managers might be fruitfully combined (as Ramirez & Bulitko combined components from both PaSSAGE and the Automated Story Director to create a new manager, PAST). By formulating experience management as the process of changing an MDP's transition function, GEM highlights the similarity between Drama Management and Dynamic Difficulty Adjustment, connecting a manager's actions in the former to changing a game's dynamics in the latter. Second, I devised and demonstrated a new way to evaluate player-specific managers (by comparing them to proportional random managers), allowing experimenters to more accurately attribute a manager's scores to the part of its behaviour that depended on its players. Finally, I designed and developed *Lord of the Borderlands* as a testbed for PaSSAGE 2, demonstrating that experience managers can be successfully integrated with tools that were built to create commercial video games.

Looking forward, a variety of opportunities for future work await, from mixing parts of existing managers to reducing experimental noise. Some of these efforts are already underway, and all of them can benefit from the work described herein.

References

Maria Arinbjarnar and Daniel Kudenko. Schemas in directed emergent drama. In Ulrike Spierling and Nicolas Szilas, editors, *Interactive Storytelling*, volume 5334 of *Lecture Notes in Computer Science*, pages 180–185. Springer Berlin Heidelberg, 2008.

R.S. Aylett, S. Louchart, J. Dias, A. Paiva, and M. Vala. Fearnot! – an experiment in emergent narrative. In Themis Panayiotopoulos, Jonathan Gratch, Ruth Aylett, Daniel Ballin, Patrick Olivier, and Thomas Rist, editors, *Intelligent Virtual Agents*, volume 3661 of *Lecture Notes in Computer Science*, pages 305–316. Springer Berlin Heidelberg, 2005.

Albert Bandura. Social cognitive theory: An agentic perspective. *Annual Review of Psychology*, 52:1–26, 2001.

Heather Barber and Daniel Kudenko. Dynamic generation of dilemma-based interactive narratives. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 2–7, Palo Alto, California, June 6-8 2007. AAAI Press.

Heather Barber, David Thue, and Christina Strong, editors. *The INTETAIN 2008 Workshop on Integrating Technologies for Interactive Stories*, Playa del Carmen, Mexico, 2008. ICST.

Joseph Bates. Virtual reality, art, and entertainment. *Presence: The Journal of Teleoperators and Virtual Environments*, 1(1):133–138, 1992.

Richard Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957.

BioWare Corp. The Aurora Neverwinter Toolset. http://nwn.bioware.com/, 2002.

BioWare Corp. The Dragon Age: Origins Toolset. http://dragonage.bioware.com/toolset/, 2011.

Marc Blank. Deadline. Infocom, 1982.

Nicholas Bonardi. Jamming with the AI musicians in Rocksmith's session mode. In *Presentations of the Vienna Game/AI Conference*, http://gameaiconf.com/recording/rocksmith-2014/, 2014. AiGameDev.com.

Michael Booth. The AI systems of Left 4 Dead. Presentation at the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference, October 2009.

Jerry M. Burger. Desire for control and the illusion of control: The effects of familiarity and sequence of outcomes. *Journal of Research in Personality*, 20(1):66 – 76, 1986.

Jacob Cohen, Patricia Cohen, Stephen G. West, and Leona S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge Academic, 3rd edition, 2002.

Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Erlbaum, Hillsdale, NJ, 2nd edition, 1988.

Andrew Csinger, Kellogg S. Booth, and David Poole. AI meets authoring: User models for intelligent multimedia. *Artificial Intelligence Review*, 8(5):447–468, 09 1994.

Bruno C. da Silva, Edwardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, USA, 2006.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, New York, 1993.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, pages 1–26, 1979.

Rui Figueiredo, Antonio Brisson, Ruth Aylett, and Ana Paiva. Emergent stories facilitated. In Ulrike Spierling and Nicolas Szilas, editors, *Interactive Storytelling*, volume 5334 of *Lecture Notes in Computer Science*, pages 218–229. Springer Berlin Heidelberg, 2008.

G.T. Graves, B.M. O'Connor, and A.C. Barker. Apparatus and method of selecting video programs based on viewers' preferences, April 25 1995. US Patent 5,410,344.

Justin Harris and R. Michael Young. Proactive mediation in plan-based narrative environments. In *Intelligent Virtual Agents*, pages 292–304, 2005.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Peter J. Huber. *Robust Statistics*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 2004.

Robin Hunicke and Vernell Chapman. AI for dynamic difficult adjustment in games. In *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*, 2004.

Darius Kazemi. Metrics and dynamic difficulty in Ritual's SiN Episodes. http://orbusgameworks .com/blog/article/70/metrics-anddynamic-difficulty-in-rituals-sin-episodes-part-1, 2008.

Hartmut Koenitz, Mads Haahr, Gabriele Ferri, and TongucIbrahim Sezen. Towards a shared vocabulary for interactive digital storytelling. In Ruth Aylett, MeiYii Lim, Sandy Louchart, Paolo Petta, and Mark Riedl, editors, *Interactive Storytelling*, volume 6432 of *Lecture Notes in Computer Science*, pages 293–294. Springer Berlin Heidelberg, 2010.

Hartmut Koenitz, Mads Haahr, Gabriele Ferri, and TongucIbrahim Sezen. Towards a unified theory for interactive digital storytelling - classifying artifacts: A workshop at ICIDS 2011. In Mei Si, David Thue, Elisabeth André, JamesC. Lester, Joshua Tanenbaum, and Veronica Zammitto, editors, *Interactive Storytelling*, volume 7069 of *Lecture Notes in Computer Science*, pages 360–361. Springer Berlin Heidelberg, 2011.

Reed Larson. Is feeling "in control" related to happiness in daily life? *Psychological Reports*, 64(3 Pt 1):75–84, Jun 1989.

Brenda Kay Laurel. *Toward the design of a computer-based interactive fantasy system*. PhD thesis, The Ohio State University, 1986.

Brenda Laurel. *Computers as Theatre*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991.

Robin Laws. Robin's laws of good GMing. Steve Jackson Games, 2001.

Greg Lee, Vadim Bulitko, and Elliot A. Ludvig. Automated story selection for color commentary in sports. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(2):144–155, June 2014.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 598–604, Bellevue, WA, USA, 2013. AAAI Press.

Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):55, 1932.

Brian Magerko. *Player Modeling in the Interactive Drama Architecture*. PhD thesis, Computer Science and Engineering, University of Michigan, 2006.

Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. *Game Developers Conference, Game Design track*, 2003.

Michael Mateas and Andrew Stern. Procedural authorship: A case-study of the interactive drama Façade. In *Digital Arts and Culture (DAC)*, Copenhagen, November 2005.

Michael Mateas. *Interactive Drama, Art, and Artificial Intelligence*. PhD thesis, Technical Report CMU-CS-02-206, School of Computer Science, Carnegie Mellon University, 2002.

Bethesda Game Studios. Skyrim. www.elderscrolls.com/skyrim, 2011.

CCP Games. Eve Online. http://www.eveonline.com/, 2003.

IBM Corp. IBM SPSS statistics for Macintosh, 2013.

Maxis Software. DarkSpore. http://darkspore.com/, 2011.

Mythos Games and MicroProse. XCOM: UFO Defense. http://www.mythosgames.com/ ufoenemyunknow.htm, 1994.

Namco. Pac-man. http://pacman.com, 1980.

Nintendo. Mario Kart 64. http://mario.nintendo.com/, 1996.

Santiago Ontañón and Jichen Zhu. The SAM algorithm for analogy-based story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 67–72. AAAI Press, 2011.

Sergio Poo Hernandez, Vadim Bulitko, and Emilie St. Hilaire. Emotion-based interactive storytelling with artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 146–152, Raleigh, North Carolina, USA, 2014. AAAI Press.

Ritual Entertainment. SiN Episodes: Emergence. http://www.sinepisodes.com/, 2006.

Rockstar North. Grand Theft Auto V. http://www.rockstargames.com/V/, 2013.

Rockstar San Diego. Red Dead Redemption. http://www.rockstargames.com/reddeadredemption/, 2010.

The MathWorks Inc. *MATLAB Version 8.1.0.604 (R2013a)*. Natick, Massachusetts, 2013.

Valve Corporation. Left 4 Dead. http://www.l4d.com/, 2008.

Valve Corporation. Steam & game stats. http://store.steampowered.com/stats/, July 2014.

Scott W. McQuiggan, Sunyoung Lee, and James C. Lester. Predicting user physiological response for interactive environments: an inductive approach. In *Proceedings of the 2nd Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 60–65. AAAI Press, 2006.

Manish Mehta, Christina Strong, and David Thue, editors. *The AAMAS 2010 Workshop on Collaborative Human/AI Control for Interactive Experiences*, number AW14, Toronto, Canada, 2010. IFAAMAS.

Mark J. Nelson, Michael Mateas, David L. Roberts, and Charles L. Isbell. Declarative optimization-based drama management in interactive fiction. *IEEE Computer Graphics and Applications*, 26(3):33–41, 2006.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

Hyacinth S. Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, 1990.

Jeff Orkin and Deb Roy. Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 385–392, 2009.

Charles Perrault. Le petit chaperon rouge. In *Histoires ou contes du temps passé, avec des moralités: Contes de ma mère l'Oye*. Paris, 1697.

Alejandro Jose Ramirez and Vadim Bulitko. Telling interactive player-specific stories and planning for it: ASD + PaSSAGE = PAST. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 173–178, Palo Alto, California, 2012. AAAI Press.

Alejandro Ramirez and Vadim Bulitko. Automated planning and player modelling for interactive storytelling. *Computational Intelligence and AI in Games, IEEE Transactions on*, pages 1–12 (in press), 2014.

Alejandro Jose Ramirez, Vadim Bulitko, and Marcia Spetch. Evaluating planning-based experience managers for agency and fun in text-based interactive narrative. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 65–71, Boston, MA, USA, 2013. AAAI Press.

MarkO. Riedl and Andrew Stern. Believable agents and intelligent story adaptation for interactive storytelling. In Stefan Göbel, Rainer Malkewitz, and Ido Iurgel, editors, *Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

Mark O. Riedl, Andrew Stern, Don Dini, and Jason Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. In H. Tianfield, editor, *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, volume 3, pages 23–42, Glasgow, 2008. SWIN Press.

David L. Roberts, Mark J. Nelson, Charles L. Isbell, Michael Mateas, and Michael L. Littman. Targeting specific distributions of trajectories in mdps. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1213–1218. AAAI Press, 2006.

David L. Roberts. Seven design challenges for fully-realized experience management. In *Intelligent Narrative Technologies*, volume WS-11-18 of *AAAI Workshops*. AAAI, 2011.

Jonathan Rowe and James Lester. A modular reinforcement learning framework for interactive narrative planning. In *Proceedings of the 6th Workshop in Intelligent Narrative Technologies*, pages 57–63, Boston, MA, USA, 2013. AAAI Press.

Jonathan P Rowe, Lucy R Shores, Bradford W Mott, and James C Lester. A framework for narrative adaptation in interactive story-based learning environments. In *Proceedings of the Third Intelligent Narrative Technologies Workshop*, pages 14–21, Monterey, CA, USA, 2010. ACM.

Stuart J. Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, third edition, 2010.

Manu Sharma, Manish Mehta, Santiago Ontañón, and Ashwin Ram. Player modeling evaluation for interactive fiction. Technical report, AIIDE 2007 Workshop on Optimizing Player Satisfaction, Palo Alto, California: AAAI Press, June 2007.

Anne Margaret Sullivan. *The Grail Framework: Making Stories Playable on Three Levels in CRPGs.* PhD thesis, UC Santa Cruz: Computer Science, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning).* The MIT Press, 1998.

Reid Swanson and Andrew Gordon. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Intelligent Interactive Systems*, 2(3):16:1–16:35, 2012.

Ivo Swartjes and Mariët Theune. Late commitment: Virtual story characters that can frame their world. Technical report, Centre for Telematics and Information Technology, Enschede, Netherlands, 2009.

Umar Syed, Robert Schapire, and Michael Bowling. Apprenticeship learning using linear programming. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, pages 1032–1039, 2008.

Nicolas Szilas, Thomas Boggini, and Paolo Petta. Sharing interactive digital storytelling technologies. In Mei Si, David Thue, Elisabeth André, JamesC. Lester, Joshua Tanenbaum, and Veronica Zammitto, editors, *Interactive Storytelling*, volume 7069 of *Lecture Notes in Computer Science*, pages 366–367. Springer Berlin Heidelberg, 2011.

Nicolas Szilas, Stefan Rank, Paolo Petta, and Wolfgang Müller. Sharing interactive digital storytelling technologies II. In David Oyarzun, Federico Peinado, R. Michael Young, Ane Elizalde, and Gonzalo Méndez, editors, *Interactive Storytelling*, volume 7648 of *Lecture Notes in Computer Science*, pages 216–216. Springer Berlin Heidelberg, 2012.

Suzanne C. Thompson and Shirlynn Spacapan. Perceptions of control in vulnerable populations. *Journal of Social Issues*, 47(4):1–21, 1991.

Suzanne C. Thompson, Craig Thomas, and Wade Armstrong. Illusions of control, underestimations, and accuracy: A control heuristic explanation. *Psychological Bulletin*, 123(2):143–161, 1998.

David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive storytelling: A player modelling approach. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 43–48, Palo Alto, California, June 6-8 2007. AAAI Press.

David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Learning player preferences to inform delayed authoring. In *Papers from the AAAI Fall Symposium on Intelligent Narrative Technologies*, volume FS-07-05, pages 158–161. AAAI Press, Arlington, Virginia, November 2007.

David Thue, Vadim Bulitko, and Marcia Spetch. PaSSAGE: A demonstration of player modelling in interactive storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 227–228. AAAI Press, 2008.

David Thue, Vadim Bulitko, and Marcia Spetch. Player Modeling for Interactive Storytelling: A Practical Approach. In Steve Rabin, editor, *AI Game Programming Wisdom*, volume IV, chapter 7, pages 633–646. Charles River Media, Boston, MA, 02/2008 2008.

David Thue, Vadim Bulitko, Marcia Spetch, and Michael Webb. Exaggerated claims for interactive stories. In *The Second Joint International Conference on Interactive Digital Storytelling*, pages 179–184, Guimarães, Portugal, December 2009. Springer Berlin / Heidelberg.

David Thue, Vadim Bulitko, Marcia Spetch, and Trevon Romanuik. Player agency and the relevance of decisions. In *The Third Joint International Conference on Interactive Digital Storytelling*, pages 210–215, Edinburgh, Scotland, November 2010. Springer Verlag.

David Thue, Vadim Bulitko, Marcia Spetch, and Michael Webb. Socially consistent characters in player-specific stories. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 198–203, Palo Alto, California, USA, 2010. AAAI Press.

David Thue, Vadim Bulitko, Marcia Spetch, and Trevon Romanuik. A computational model of perceived agency in video games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 91–96, Palo Alto, California, USA, 2011. AAAI Press.

David Thue. Player-informed interactive storytelling. Master's thesis, University of Alberta, 2007.

Zach Tomaszewski and Kim Binsted. Demeter: An implementation of the marlinspike interactive drama system. In *AAAI Symposium on Intelligent Narrative Technologies II*, pages 133–136, Palo Alto, California, March 2009. AAAI Press.

John Wilder Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

Anders Tychsen. Tales for the many: Process and authorial control in multi-player role-playing games. In Ulrike Spierling and Nicolas Szilas, editors, *Interactive Storytelling*, volume 5334 of *Lecture Notes in Computer Science*, pages 309–320. Springer Berlin Heidelberg, 2008.

Ubisoft. Rocksmith 2014. http://rocksmith.ubi.com/, 2013.

Ivar E. Vermeulen, Christian Roth, Peter Vorderer, and Christoph Klimmt. Measuring user responses to interactive stories: towards a standardized assessment tool. In *The Third Joint International Conference on Interactive Digital Storytelling*, pages 38–43, Edinburgh, Scotland, 2010. Springer-Verlag.

Noah Wardrip-Fruin, Michael Mateas, Steven Dow, and Serdar Sali. Agency reconsidered. In *Proceedings of the Digital Games Research Association 2009 Conference (DIGRA 09)*, 2009.

Allan Weallans, Sandy Louchart, and Ruth Aylett. Distributed drama management: beyond double appraisal in emergent narrative. In *Proceedings of the 5th international conference on Interactive Storytelling*, ICIDS'12, pages 132–143, Berlin, Heidelberg, 2012. Springer-Verlag.

P. Weyhrauch. *Guiding Interactive Drama*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.

Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on*, 2(3):147–161, July 2011.

Georgios N. Yannakakis, Henrik Hautop Lund, and John Hallam. Modeling children's entertainment in the playware playground. In *Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games (CIG06), University of Nevada, Reno, campus in Reno/Lake Tahoe, 22-24 May, 2006*, pages 134–141, 2006.

R. Michael Young, Mark O. Riedl, Mark Branly, and Arnav Jhala. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):54–70, 2004.

Hong Yu and Mark O. Riedl. A sequential recommendation approach for interactive personalized story generation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 1 of *AAMAS '12*, pages 71–78, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

Alexander E. Zook and Mark O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 93–98. AAAI Press, 2012.

Alexander E. Zook and Mark O. Riedl. Temporal game challenge tailoring. *Computational Intelligence and AI in Games, IEEE Transactions on*, pages 1–11 (in press), 2014.

Appendix A

User Study Materials

A.1 Survey 1

See pages 116 to 117.

A.2 Survey 2

See pages 118 to 121.

A.3 PaSSAGE Study Materials

See pages 122 to 124.

A.4 PaSSAGE 2 Study Materials

See pages 125 to 126.

Game Story Evaluation

Age (in years): ____ **Gender**: M F

For the following section, please choose a **number** between 1 and 7 to describe your opinion regarding the game story that you just experienced. Please circle only one number per statement. If you agree with the term on the left, circle 1. If you agree with the term on the right, circle 7. If you feel that your opinion fits somewhere between these terms, circle a number between 1 and 7.

***** Carefully read each pair of terms before circling your answer *****

In comparison to an average video game of similar length that you've played in the past (or your expectation of one), how enjoyable was your game experience?

1	2	3	4	5	6	7
(Less fun)						(More fun)

I felt as though my actions were influencing the story:

1	2	3	4	5	6	7
(Never)						(Always)

Overall, my interest in playing this game again is:

1	2	3	4	5	6	7
(Low)						(High)

In an average week, the amount of time that I spend playing video games is:

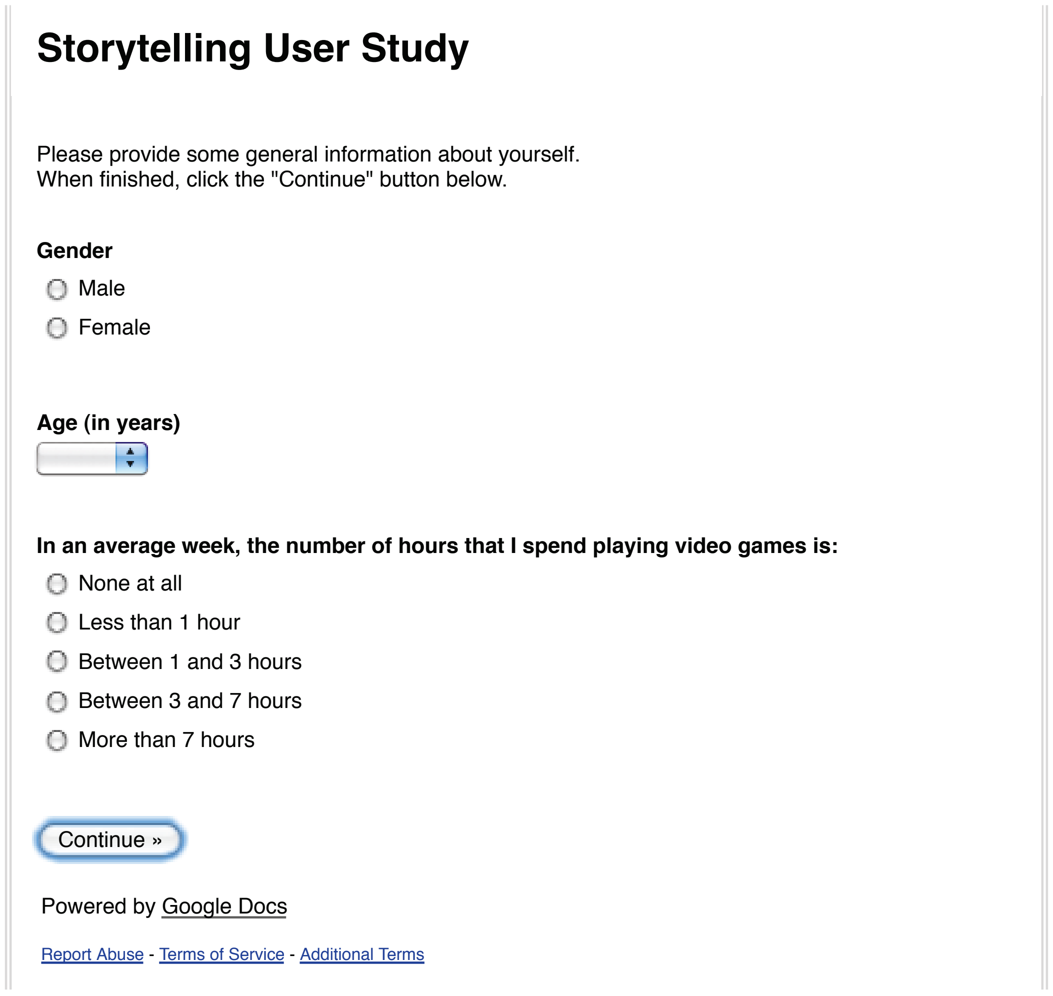
1	2	3	4	5	6	7
(None)	(0-1 hr)	(1-3)	(3-6)	(6-9)	(9-12)	(>12 hrs)

In my opinion, this game was:

1	2	3	4	5	6	7
(Engaging)						(Shallow)
1	2	3	4	5	6	7
(Plausible)						(Implausible)
1	2	3	4	5	6	7
(Interesting)						(Boring)
1	2	3	4	5	6	7
(Predictable)						(Surprising)
1	2	3	4	5	6	7
(Fascinating)						(Ordinary)
1	2	3	4	5	6	7
(Repetitive)						(Varied)
1	2	3	4	5	6	7
(Hard to follow)						(Easy to follow)
1	2	3	4	5	6	7
(Creative)						(Conventional)

Comments (optional): Please write any comments you may have on the reverse of this page.

Comments (optional):



Storytelling User Study

Please provide some general information about yourself.
When finished, click the "Continue" button below.

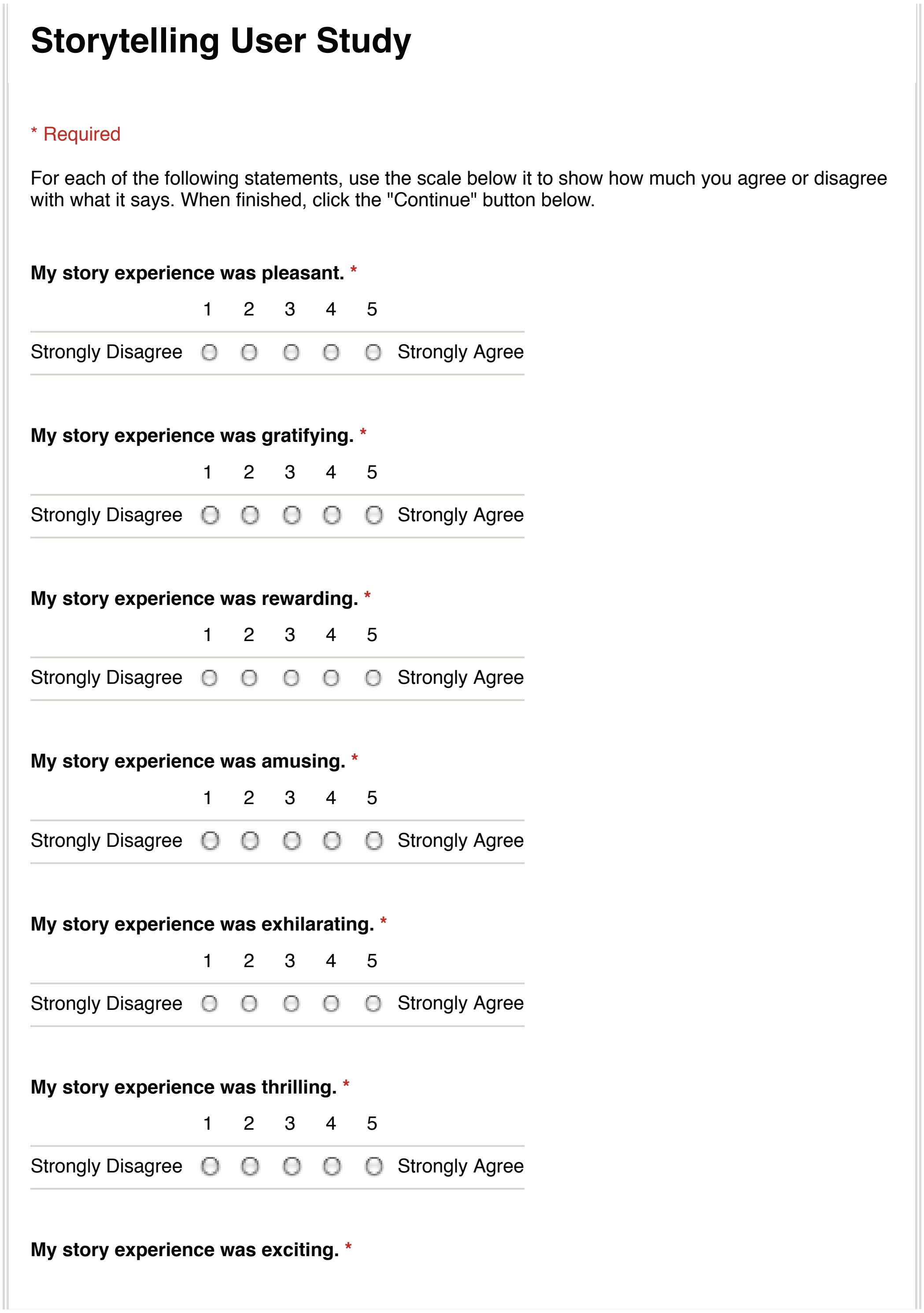
Gender

- ◯ Male
- ◯ Female

Age (in years)

In an average week, the number of hours that I spend playing video games is:

- ◯ None at all
- ◯ Less than 1 hour
- ◯ Between 1 and 3 hours
- ◯ Between 3 and 7 hours
- ◯ More than 7 hours



Storytelling User Study

* Required

For each of the following statements, use the scale below it to show how much you agree or disagree with what it says. When finished, click the "Continue" button below.

My story experience was pleasant. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was gratifying. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was rewarding. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was amusing. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

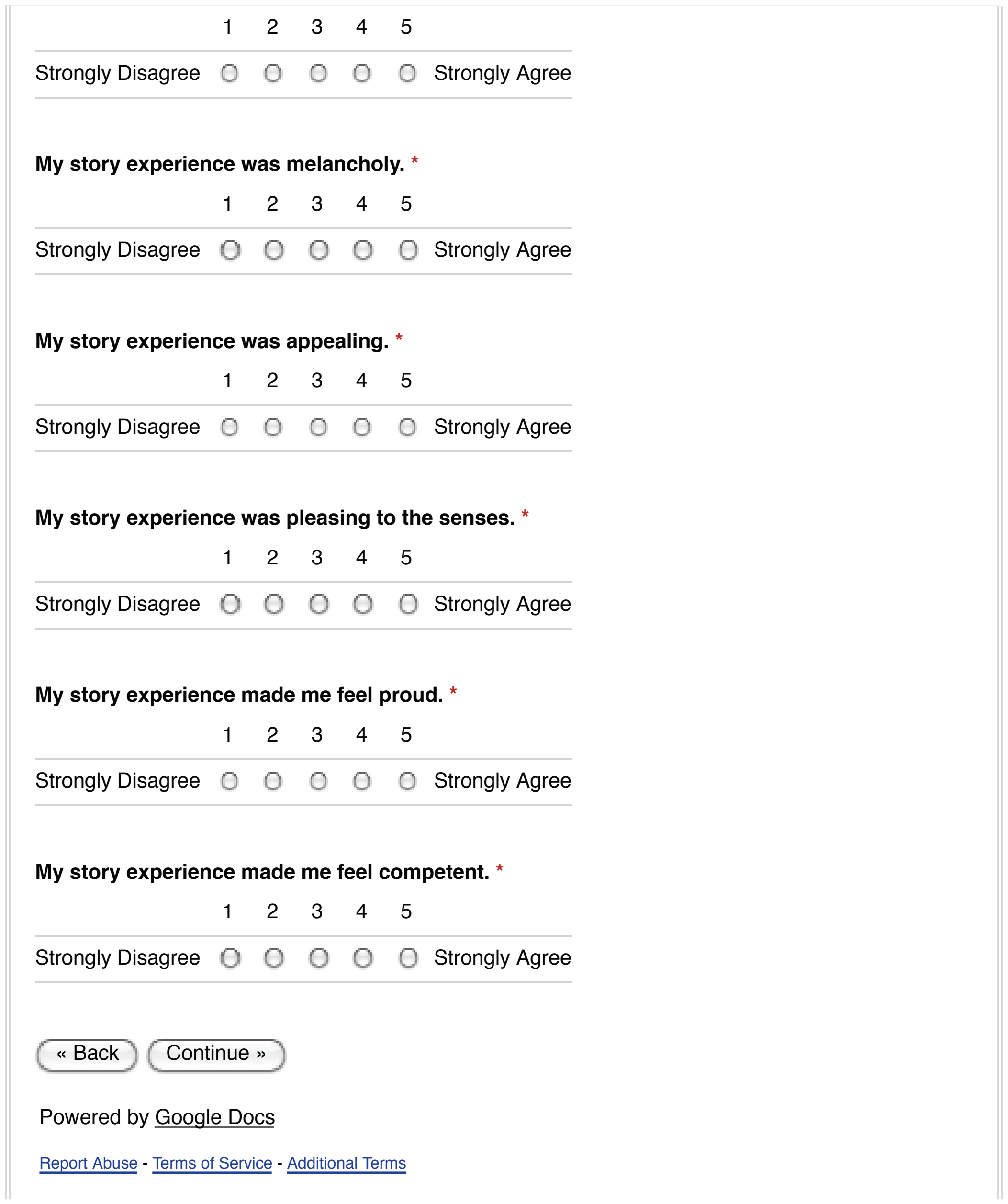
My story experience was exhilarating. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was thrilling. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was exciting. *



	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was melancholy. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was appealing. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience was pleasing to the senses. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience made me feel proud. *

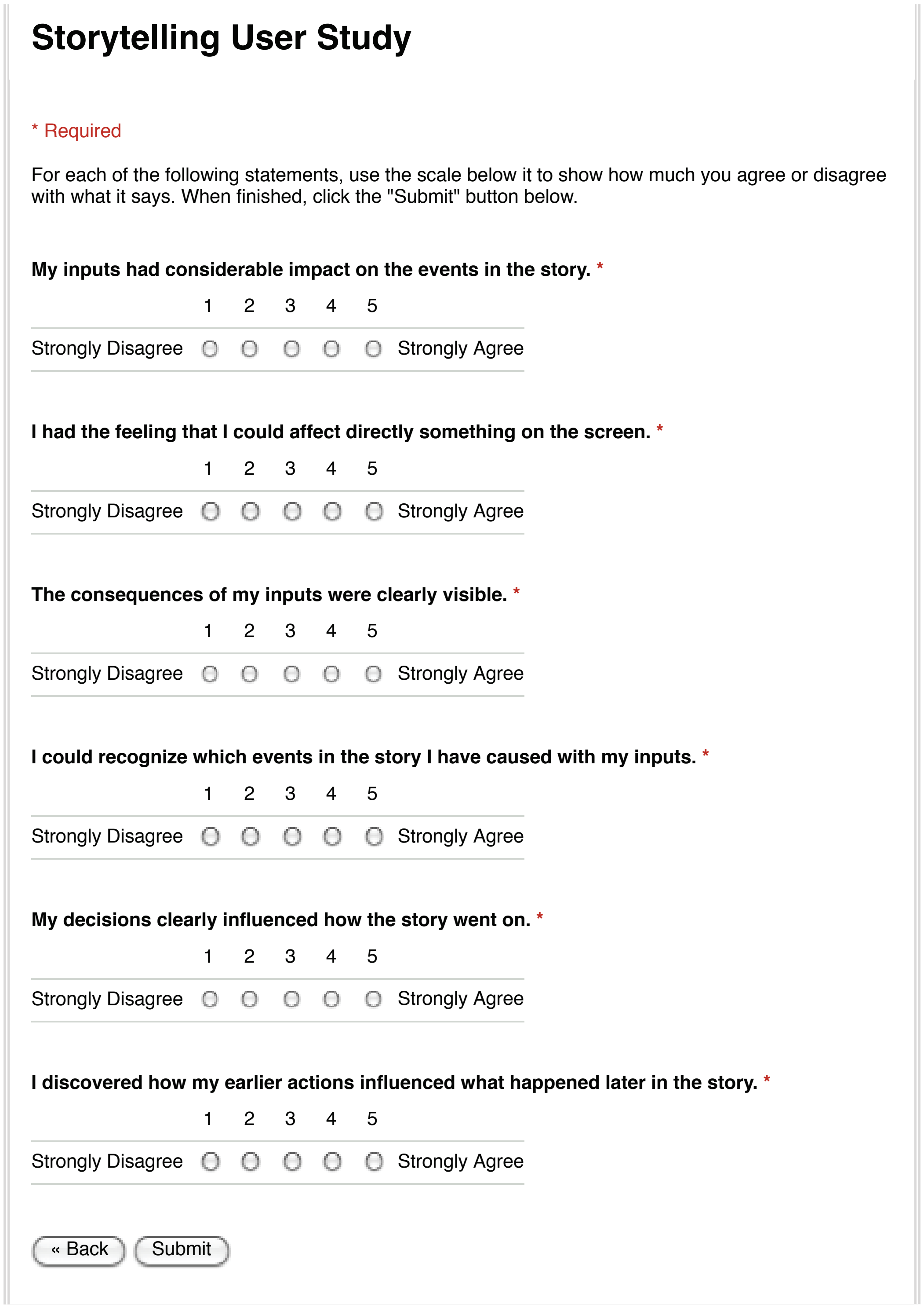
	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My story experience made me feel competent. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Powered by Google Docs

Report Abuse - Terms of Service - Additional Terms



Storytelling User Study

* Required

For each of the following statements, use the scale below it to show how much you agree or disagree with what it says. When finished, click the "Submit" button below.

My inputs had considerable impact on the events in the story. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I had the feeling that I could affect directly something on the screen. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

The consequences of my inputs were clearly visible. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I could recognize which events in the story I have caused with my inputs. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

My decisions clearly influenced how the story went on. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

I discovered how my earlier actions influenced what happened later in the story. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Introduction and Participants' Consent Form

Introduction. Welcome! You are invited to participate in a research study being conducted by David Thue and Dr. Vadim Bulitko of the Department of Computing Science and Dr. Marcia Spetch of the Department of Psychology, from the University of Alberta. The purpose of this study is to evaluate the quality and entertainment value of several *interactive stories* (video game experiences in which the player chooses the actions of the story's main character, thereby adjusting the course of the story).

Your participation. Your participation in this study involves experiencing a short interactive story by playing a computer video game for about 30 minutes. The events that form the story may include animated violence with minimal blood. Before beginning the story, you will be presented with a short set of instructions for interacting with the game environment, and given the opportunity to practice for 5 to 10 minutes. Following your completion of the story, you will be asked to fill out a survey ranking the game across several measures including whether you found the story interesting or boring, surprising or predictable, etc.

Your rights. Your decision to participate in this study is entirely voluntary and you may decide at any time to withdraw from the study. If you choose to participate, you may skip any items you do not wish to answer. Responses made by individual participants on the questionnaires will remain confidential, and your name will not appear on the questionnaire or be associated with your responses in any way. Questionnaires will be identified only by a researcher-assigned code number, for the purpose of associating survey forms with the particular story that the participant experienced. Only researchers associated with the project will have access to the questionnaires. The results of this study may be presented at scholarly conferences, published in professional journals or books, or presented in class lectures. All data presented will be anonymous. The data will be securely stored by (David Thue) for a minimum of five years. If you choose to decline or discontinue your participation, we will accept your decision without question and without penalty, i.e., you will still receive course credit.

Benefits and risks. There are no foreseeable risks to this study, but if any risks should arise, the researcher will inform the participants immediately. If you should experience any adverse effects, please contact David Thue and/or Dr. Vadim Bulitko immediately.

Contact information. If you have any questions or comments on the study, or if you wish a clarification of rights as a research participant, you can contact David Thue or the Human Research Ethics Committee at the number and address below.

David Thue	***Vadim Bulitko***, Ph.D.	***Chair***
Ph.D. Candidate	Associate Professor	Arts, Science, and Law Research
Department of Computing Science	Department of Computing Science	Ethics Board
University of Alberta	University of Alberta	Faculty of Arts
Edmonton, AB T6G 2E8	Edmonton, AB T6G 2E8	University of Alberta
(780) 492-2821	(780) 492-3854	(780) 492-4224
dthue@cs.ualberta.ca	bulitko@ualberta.ca	ASLREBAdministrator@ualberta.ca

Signatures. Please sign below to indicate that you have read and understood the nature and purpose of the study. Your signature acknowledges the receipt of a copy of the consent form as well as indicates your willingness to participate in this study.

Participant's Signature ______________________ Date ____________

Researcher's Signature ______________________ Date ____________

Debriefing

Evaluating Entertainment of Recurring Characters in Adaptive Interactive Stories

Thank you for participating in this study! Your time and effort have been very valuable to us. Video games are an increasingly popular form of entertainment, but little is known about why people play video games and what gratifications they receive from them. Our research investigates two questions: 1) whether personal agency influences the enjoyment experienced from playing a game, and 2) whether the socially consistent behaviour from a game story's characters (e.g., a character that the player insulted may be less likely to help the player later on) can make that story more engaging for its player. Specifically, we wonder: 1) whether a storytelling game that adapts to a player's behavior would be rated as being more enjoyable than one whose events occurred independently of the player's behaviour (randomly, in this case), and 2) whether a storytelling game whose characters behave in a socially consistent manner would be rated as being more engaging than one in which no social consistency (or inconsistency) is displayed. A great deal of previous research by Seligman and others has shown that a lack of personal control over aversive events is stressful, but little research has addressed the extent to which personal agency enhances enjoyment. To examine this, we created an automated storytelling system in which the story could automatically change depending on your actions and decisions in the game environment.

Our hypotheses are twofold: 1) players who experience the automatically-adapting story will rate the game's entertainment value more highly than players who experience the random story; and 2) players who experience the story with socially consistent character behaviour will rate the game as being more engaging than the game without such consistency. It was necessary to withhold the information that the game may adapt its story based on your actions because your rating of the game's entertainment value might have been biased by believing that an adaptive, "intelligent" system was enabled (the placebo effect).

To test our first hypothesis, our independent variable is the presence or absence of adaptive learning by the game. If you were in control condition A, the automatic adaptive system was turned off, and story events were chosen at random. If you were in experimental condition A, the game automatically observed the actions that you took, and tried to learn your preferred way of playing; based on this learned model, the game then chose events to occur that would allow you to continue to play in your preferred way, while still satisfying the overall structure of a heroic journey through a fantasy world. Thus, the experimental condition provided personal agency over the events in the game whereas the control condition did not. Our primary dependent variable was your rating of how enjoyable the game was on the questionnaire.

To test our second hypothesis, our independent variable is the presence or absence of socially consistent character behaviour during the game. If you were in control condition B, no consistency was displayed between player/character interactions and subsequent character behaviours. If you were in experimental condition B, the game automatically selected characters for subsequent story events based on your previous interactions with them during the game; characters who were slighted by the player might appear later on as opponents, while characters with whom the player had good dealings might appear to save the day. Thus, the experimental condition provided consistency between the player's interactions with characters and their behaviours later in the story, whereas the control condition did not. Our primary dependent variable was your rating of how engaging the game was on the questionnaire.

The results of this research will further understanding of the importance of personal agency and will provide knowledge about the factors that influence gratification and enjoyment of storytelling games. This information will be of theoretical interest to researchers in both psychology and computer science, and will have applied value for researchers in computer science as well.

Thanks very much for participating. Without the help of people like you, we couldn't answer most important scientific questions in psychology. You've been a great help. Do you have any questions that I can answer right now? If you have any questions, later on, about the study, please contact David Thue via either phone (492-2821) or e-mail (dthue@cs.ualberta.ca) or if you have general questions, contact Dr. Tom Spalding (Director, Research Participation) at rpdirect@ualberta.ca or 492-7778, or Sharon Randon (Research Participation Coordinator) at rescred@ualberta.ca or 492-5689. Please don't tell other people about what we had you do here because other students in the class may participate in this study.

Introduction and Participants' Consent Form

Introduction. Welcome! You are invited to participate in a research study being conducted by David Thue and Dr. Vadim Bulitko of the Department of Computing Science and Dr. Marcia Spetch of the Department of Psychology, from the University of Alberta. The purpose of this study is to evaluate the quality and entertainment value of several *interactive stories* (video game experiences in which the player chooses the actions of the story's main character, thereby adjusting the course of the story). **Each session of this study will last for at most one hour and fifty minutes.**

Your participation. Your participation in this study involves experiencing a short interactive story by playing a computer video game for about 90 minutes. The events that form the story may include animated violence with minimal blood. Before beginning the story, you will be presented with a short set of instructions for interacting with the game environment, and given the opportunity to practice for 5 to 10 minutes. Following your completion of the story, you will be asked to fill out a survey ranking the game across several measures. **Your participation in this study is worth 4% toward your course mark.**

Your rights. Your decision to participate in this study is entirely voluntary and you may decide at any time to withdraw. If you choose not to participate or withdraw after you have begun, but would like your 4% credit for participation, you may complete an alternative educational activity. In this case, you will be given a short article to read on the decisions that storytellers make while telling a story. You will be asked to answer on paper a few questions about the article. The time it takes to complete this assignment will take no longer than the time it takes to participate in this study. Your decision not to participate will not affect access to services from the University of Alberta. Your survey responses will remain confidential and anonymous, and our data file will NOT contain any personal identifiers (i.e., names or student ID numbers). Survey forms will be identified only by a researcher-assigned code number, for the purpose of associating them with the particular story that the participant experienced. Only researchers associated with the project will have access to the questionnaires. The results of this study may be presented at scholarly conferences, published in professional journals or books, or presented in class lectures. All data presented will be anonymous. The data will be securely stored by (David Thue) for a minimum of five years.

Benefits and risks. There are no foreseeable risks to this study, but if any risks should arise, the researcher will inform the participants immediately. If you should experience any adverse effects, please contact David Thue and/or Dr. Vadim Bulitko immediately.

Contact information. If you have any questions or comments on the study, or if you wish a clarification of rights as a research participant, you can contact David Thue or the Human Research Ethics Committee at the number and address below.

David Thue	***Vadim Bulitko***, Ph.D.	***Chair***
Ph.D. Candidate	Associate Professor	Arts, Science, and Law Research
Department of Computing Science	Department of Computing Science	Ethics Board
University of Alberta	University of Alberta	Faculty of Arts
Edmonton, AB T6G 2E8	Edmonton, AB T6G 2E8	University of Alberta
(780) 492-2821	(780) 492-3854	(780) 492-4224
dthue@cs.ualberta.ca	bulitko@ualberta.ca	ASLREBAdministrator@ualberta.ca

Signatures. Please sign below to indicate that you have read and understood the nature and purpose of the study. Your signature acknowledges the receipt of a copy of the consent form as well as indicates your willingness to participate in this study.

_______________________________________ ____________

Participant's Signature Date

_______________________________________ ____________

Researcher's Signature Date

Debriefing

Evaluating Personal Agency in Adaptive Interactive Stories

Thank you for participating in this study! Your time and effort have been very valuable to us. Video games are an increasingly popular form of entertainment, but little is known about why people play video games and what gratifications they receive from them.

Our research in this study investigates personal agency and enjoyment in the context of a dynamically managed interactive story. Specifically, we wonder if customizing the outcomes of players' decisions on an individual basis can increase the feelings of agency and enjoyment that they derive from playing. Previous research by Seligman has shown that a lack of personal control over aversive events is stressful, and Thompson has proposed that feelings of control are stronger when one's decisions lead to desirable outcomes (versus undesirable outcomes). To test Thompson's proposal, we created an automated storytelling system in which the story could automatically change depending on your actions and decisions in the game environment.

Our hypothesis is that players who experience the automatically-adapting story will rate both the game's entertainment value and their feelings of agency while playing more highly than players who experience a story that does not adapt. It was necessary to withhold the information that the game may adapt its story based on your actions because your ratings might have been biased by believing that an adaptive, "intelligent" system was enabled (the placebo effect).

To test our hypothesis, our independent variable is the presence or absence of adaptive learning by the game. If you were in the control condition, the automatic adaptive system was turned off, and story events were chosen to be the same as those that were seen by a random previous player. If you were in the experimental condition, the game automatically observed the actions that you took, and tried to learn your preferred way of playing; based on this learned model, the game then chose which story events would occur, trying to dynamically make your decisions lead to situations that were desirable for you. Thus, the experimental condition provided desirable outcomes for player decisions in the game whereas the control condition did not. Our primary dependent variables were your ratings on the questionnaire of how enjoyable the game was, as well as how much agency you felt.

The results of this research will further understanding of the importance and causes of personal agency, and will provide knowledge about the factors that influence gratification and enjoyment of storytelling games. This information will be of theoretical interest to researchers in both psychology and computer science, and will have applied value for researchers in computer science as well.

Thanks very much for participating. Without the help of people like you, we couldn't answer most important scientific questions in psychology. You've been a great help. Do you have any questions that I can answer right now? If you have any questions, later on, about the study, please contact David Thue via either phone (492-2821) or e-mail (dthue@cs.ualberta.ca) or if you have general questions, contact Sharon Randon (Research Participation Coordinator) at rescred@ualberta.ca or 492-5689. Please don't tell others about what we had you do here because other students in the class may participate in this study.

Appendix B

Related Publications

The following citations describe the various publications that have arisen from this work.

David Thue, Vadim Bulitko, and Marcia Spetch. Making stories player-specific: Delayed authoring in interactive storytelling. In *The First Joint International Conference on Interactive Digital Storytelling (ICIDS)*, pages 230–241, Erfurt, Germany, 2008. Springer Berlin / Heidelberg.

David Thue, Vadim Bulitko, and Marcia Spetch. PaSSAGE: A demonstration of player modelling in interactive storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 227–228. Palo Alto, California, 2008. AAAI Press.

David Thue, Vadim Bulitko, and Marcia Spetch. Player modeling for interactive storytelling: A practical approach. In Steve Rabin, editor, *AI Game Programming Wisdom*, volume IV, chapter 7, pages 633–646. Boston, Massachusetts, 2008. Charles River Media.

David Thue, Vadim Bulitko, Marcia Spetch, and Michael Webb. Exaggerated claims for interactive stories. In *The Second Joint International Conference on Interactive Digital Storytelling (ICIDS)*, pages 179–184, Guimarães, Portugal, 2009. Springer Berlin / Heidelberg.

David Thue, Vadim Bulitko, Marcia Spetch, and Trevon Romanuik. Player agency and the relevance of decisions. In *The Third Joint International Conference on Interactive Digital Storytelling (ICIDS)*, pages 210–215, Edinburgh, Scotland, 2010. Springer Verlag.

David Thue, Vadim Bulitko, Marcia Spetch, and Michael Webb. Socially consistent characters in player-specific stories. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 198–203, Palo Alto, California, 2010. AAAI Press.

Mark Riedl, David Thue and Vadim Bulitko. Game AI as storytelling. In book: *Applied Research in Artificial Intelligence for Computer Games*, pages 125–150. Springer USA.

David Thue, Vadim Bulitko, Marcia Spetch, and Trevon Romanuik. A computational model of perceived agency in video games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 91–96, Palo Alto, California, USA, 2011. AAAI Press.

David Thue and Vadim Bulitko. Procedural game adaptation: Framing experience management as changing an MDP. In *Proceedings of the 5th Workshop on Intelligent Narrative Technologies*, pages 44–50, Palo Alto, California, 2012. AAAI Press.

David Thue, Vadim Bulitko, and Howard J. Hamilton. Implementation cost and efficiency for AI experience managers. In *Proceedings of the 6th Workshop on Intelligent Narrative Technologies*, pages 97–100, Boston, Massachusetts, 2013. AAAI Press.

Vadim Bulitko, Greg Lee, Sergio Poo Hernandez, Alejandro Ramirez, and David Thue. Techniques for AI-driven experience management in interactive narratives. In Steve Rabin, editor, *Game AI Pro*. Volume II, in press, 2014. CRC Press.