12-2017

Video Game Evaluation Based on Player Decision Cycles

Adam Wentworth
Clemson University

VIDEO GAME EVALUATION BASED ON PLAYER DECISION CYCLES

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

---

by
Adam Wentworth
December 2017

Accepted by:
Dr. Brian A. Malloy, Committee Chair
Meihua Qian
Eric Patterson

Abstract

In this thesis we describe a three pronged framework that captures the important aspects of player engagement in a video game. The focus of the framework is the *Player Decision Cycle*, which refers to the overlapping decisions that players make during game play. These various decisions must be made many times, but each type of decision has to be made at different rates. These decisions interact to produce the pacing of the gameplay experience. To illustrate the framework we have designed and implemented two prototype RPGs. These two RPGs are *Dragon Mist* and *Radium Seas*. These games exemplify adherence to the Player Decision Cycle framework presented in this thesis.

Table of Contents

Table of Contents (Continued)

Page

List of Figures

Figure Page

List of Figures (Continued)

Page

List of Figures (Continued)

Page

Page

Chapter 1

Introduction and Motivation

The video game industry has grown from the casual pastime of few to a large part of the entertainment industry. In 2016, video game revenues surpassed both music and movie industry revenues [17]. The gaming industry has grown so quickly that by the end of 2017, global video game revenues are projected to exceed 91 billion dollars [13]. During much of it's history, video game play has been dominated by dedicated players who were willing to spend hours crafting a complex story by playing role playing games (RPGs) or honing their hand-eye coordination for high speed first person shooters (FPSs) [14]. Historically, RPGs have been a staple of the games industry, for example with the modern popularity of games like the Elder Scrolls series and Fallout series by Bethesda [20; 23], The Witcher Series by Bioware [24], and other video games including the Pirates of the Burning Seas by Flying Labs Software [22].

Although some researchers have argued, contrary to the opinion of many, that video games themselves are educational [8; 9; 10; 12], the rising popularity of video games has not led to a similar rise in video games that teach concepts traditionally taught in levels ranging from Kindergarten through 12th grade. Some scholars resolve never to use game-based learning because high quality education games are unavailable [5; 16]. Other researchers and gamers argue that educational games, sometimes referred to as serious games, are downright boring. For example, Fletcher recommends that "The new rule for education video games is: Don't be Boring" [6]; Peters recommends "Never play a video game that's trying to teach you something" [15]; and finally, Becker suggests that education games are boring because "Making good video games is hard; Designing good instructional interventions is also hard" [1].

One explanation for the difficulty of constructing engaging education video games is the difficulty in specifying those aspects of a game that actually make it engaging. Several

researchers have attempted to describe the features or measures of education games that make them engaging. For example, Gee maintains that learning is fun and that designers of "good" games "have hit on profoundly good methods of getting people to learn and to enjoy learning" [10, p. 2]. He further describes thirteen principles of learning that good game developers incorporate into their successful games, for example permitting gamers to customize their games, well-ordered problem solving, providing safe havens or "sand boxes", and the development of skills as strategies. However, although Gee effectively describes the successful learning strategies exploited by good game developers, he fails to specify those features or aspects of a game that make it engaging; moreover, there are games that have not incorporated one or more of the thirteen principles that he describes yet they have been successful.

In this thesis we describe a three pronged framework that captures the important aspects of player engagement in a video game by separating interactive game elements into larger groups or *cycles*. These three cycles are *Continuous Decision Cycles*, *Deliberate Decision Cycles*, and *Play Framing Decision Cycles*. These cycles ask the player to interact with various game elements as groups, and the pacing of these decisions interact to produce the overall game experience. To illustrate the framework, we have used it to analyze popular games and used it to to design two prototype RPGs, *Dragon Mist* and *Radium Seas*.

In the next chapter, we provide background about the various genres of video games, especially the three genres that best fit this discussion. In Chapter 3 we review the work that relates to ours. In Chapter 4 we describe our three pronged framework for evaluating engagement. Chapter 5 illustrates the use of this framework in our design of the Skyrim mod, Dragon Mist, and the game, Radium Seas, that we have developed for this thesis. In Chapter 6 we describe our implementation of the Skyrim Mod, which uses the Bethesda Creation Kit, and in Chapter 7 we describe our implementation of Radium Seas, which uses the Unreal Engine version 4. Finally, in Chapter 8 we draw conclusions and describe our future work.

Chapter 2

Background

In this chapter, we define terms and describe concepts and techniques that we use in the design and implementation of our own game, *Radium Seas*, and a mod for Skyrim, *Dragon Mist*. We begin by defining a player, or avatar, and a non-playable character (NPC). Next, we define terms that are used heavily in the description of the process of game creation: game engines (specifically the Unreal Engine 4 and the Creation Engine), game assets, scripting, and game themes. We then define three game genres that we discuss in this thesis: The role playing game (RPG), the action game, and the open world game.

2.1 An Overview of Game Elements

> A game is a series of meaningful choices.
>
> *Sid Meier*

For the purposes of this thesis, we will use the term *game* to describe an interactive experience in which a player or players interact with a computer program. This computer program is a collection of many *game elements* that prompt player actions, reactions, and provide feedback about the changing state of the game. Managing the player's interest is a common function of all game elements.

The *educational game* is a specific type of game used to further an educational goal. Educational games are often synthesized from elements borrowed from more traditional commercial game designs. These games take many forms and have no truly universal elements. These games also are rarely made for a purely educational purpose and are often made with commercial purposes in mind.

A *player* in a game is, in many cases, the most complicated addition to the game. The player sits outside of the game and attempts to interact with the system for a wide variety

of reasons. Player motivations are discussed in detail in Cowley et al. [3] in 2.2 Player Modeling - The User Segment. The player is usually embodied in the game world by a specific object that the gamer can control. This object is called an *avatar*. The avatar can take many forms and can even be simply a camera with no actual physical appearance in the world.

An *NPC* is any of a wide variety of computer controlled objects in a world and the behavior of an NPC is typically clearly specified, possibly dependent on the state of the game. This behavior can be as simple as an enemy NPC that moves in a simple line and must be dodged or killed by the player, or an ally that will interact with the player in dialogue and follow them into battle.

A *game engine* is the framework around which a game is created. It manages the many different elements of a game and organizes them into a whole by providing basic functionality. This functionality handles everything from the types of code it supports to the type of graphics it allows and many other specifics as well. In this thesis we will be discussing two specific game engines: The Unreal Engine 4 (generally called UE4) by Epic Games, and the Creation Engine by Bethesda Games [7; 19].

The UE4 was originally developed for a specific game in 1998 called Unreal, it has since been broadened into a general purpose engine capable of supporting many game types and genre. It utilizes C++ code but supports rapid iteration through a node based scripting system called *Blueprint*. It also fosters a network of learning videos and asset distribution through its launch program called the *Epic Games Launcher* as well as in the broader online community.

The *Creation Engine* is a game engine developed by Bethesda Game Studios in 2011 for The Elder Scrolls V: Skyrim [19]. The engine is proprietary to Bethesda Game Studios, but a tool was released to the online community with the release of Skyrim called the *Creation Kit*. This tool is designed to allow the players of Skyrim to mod the game by adding new landscapes, enemies, weapons, and objectives to the game. It supports scripting through the use of a language called Papyrus. Game assets are a broad collection of files used in

the creation of a game. A list includes meshes used to define a three dimensional object, image files used to define textures, and text files used to hold individual pieces of code. It can also include meta assets that help sort the use of other game assets. These meta objects can organize groups of textures into objects called materials that define how a surface will interact with light in complex situations, organize materials with meshes to give the outer surface of a three dimensional object specific looks, and organize those with animation and artificial intelligence behaviors into objects called *actors*.

Scripting is a specific type of programming designed to add functionality to large pieces of software. In game development, this type of programming leverages game engine functionality into specific game events. Generally, scripts are relatively short pieces of code. In UE4 this functionality is handled through the Blueprint system. In the Creation Kit this is handled through the use of a scripting language called *Papyrus*.

Video game worlds are designed thematically. Two games with similar gameplay elements can induce very different reactions from the player based on how those elements are framed. The Elder Scrolls V: Skyrim and Fallout IV are, from a gameplay perspective, very similar games [18]. They are both open world RPGs with action elements. However, they look and play very differently because they have different themes. Skyrim is themed as a fantasy world of magic and dragons, while Fallout IV is themed as a post-apocalyptic retro-futuristic world of radiation and mutants. The effect of the theme on the gameplay elements filters down to all levels of game asset design. Therefore, the theme of a video game is a lens for both the player and the designer.

The term *mod* is used by the gaming community to describe the action of taking an existing game and modifying it in some way, for example to add new assets, to add new quests, or possibly add new features. This approach utilizes existing game assets to achieve a new outcome. The level at which the pre-existing assets are used varies greatly between mods. As the amount of new assets and features increases, a mod can bridge the gap into a full and independent new game. Examples include The Stanley Parable, Counter-Strike, and Defense of the Ancients 2 (sometimes referred to colloquially as DOTA 2).

2.2 Role Playing Games

A *role playing game*, or RPG, can be defined as a specific set of game elements that focus heavily on the player assuming the role of a specific character or characters over the course of a game's narrative. This heavy investment in characterization and story narrative is usually supported by many game elements designed to increase the player's feeling of ownership over those characters and their story. Generally these game elements are manifested in only a few standard ways. The player is offered choice in dialogue options to define their character's story, such as whether or not to perform certain tasks or what in-story groups to join. This often takes the form of quests that guide the player through the game and serve to drive the narrative. The player is offered choices in character design, often called leveling up, which allows the player to alter the way their character is allowed to behave in the rest of the game. This leveling up is frequently accompanied by other improvements, such as skill improvements that generally improve the way that the player can interact in the game world. For example, the player may now be able to defeat an enemy that was too difficult before leveling up. Similarly, the player may be offered new dialog options as their character improves their speech or bartering ability.

The player also has the task of managing items to better customize their character. Items, usually found or purchased during the course of the game, can be used to alter how the player can interact with the game in the short term. Where leveling up is a long term decision, equipping items allows the player to alter their character in the short term.

2.3 Action Games

The action game can here be defined as a group of game elements that focus on short term player engagement. These challenges usually rely on reaction time and avatar control in changing situations. Game elements that support the action game are designed to push the player in the short term and create tension through the management of complex, fast changing situations. The player generally engages in some form of combat with NPCs.

Usually this action element involves avatar movement, combat tactics, and management of health. The player also must engage the level by overcoming obstacles, avoiding traps, or solving puzzles. Usually this action element involves the player observing their avatar's surroundings to notice dangerous areas, or clues to solve the puzzle. Dangers must then be avoided or exploited for the player's advantage, puzzles solved to continue in the level.

Obstacles or traps generally present dangerous situations but can be rendered neutral or used to the player's advantage. For example, enemies can be lured into a spike trap or an oil pit and be defeated by exploiting the trap. The trap always activates, regardless of whether the player or an NPC interacts with the trap.

The player also must manage their avatar's abilities. To overcome the previous elements, the player has a suite of abilities to manage. These abilities can be consistent, such as their avatar's ability to move in space. These abilities can also be inconstant, such as an ability to move more quickly for a short period of time or do a large amount of damage to an enemy or enemies very rarely. Deciding how and when to use these abilities is designed to engage the player in short term challenges.

2.4 Open World Games

The open world game is a relatively new genre of video game. It is difficult to define, as it generally involves elements of many other game design types. However, certain game elements appear to be unique to the genre and will be how we define it for the course of this thesis. These elements focus on the design of the space that the player can move in, as well as how they are directed to move through it, to increase the feeling of player investment in a story as well as developing a unique story internal to the player in response to the game. The space the player can move through in this type of game is generally large and free of boundaries. This space is often referred to as the *map*. The map is filled with many obstacles and NPCs. Challenges for the player develop organically in response to their own movement through this map.

The map is generally subdivided into areas often called *zones*. These zones generally contain challenges of a similar type, allowing the player to choose what zone to explore based on the first encounters they have in an area. If they find themselves too challenged, they may not engage with an area until they feel more accomplished.

Zones are further divided into more specific *areas*. These areas encompass thematically similar challenges. A town might contain social challenges, further story development, or be a place to heal. A dungeon might contain traps, monsters, or great rewards. Grouping challenges into broadly visible areas further drives the player's ability to choose how they wish to engage with the game.

Areas can be further divided into *rooms*. Rooms usually contain a specific challenge to be overcome. This might be a group of enemies or a specific NPC who has information for the player. Having individual rooms be relatively separate from the rest of a zone allows the player to approach each challenge with a sense of agency as they have chosen it out of all of the others in the map as a whole.

Chapter 3

Related Work

In this chapter we review the research that relates to the topics described in this thesis. In Cowley et al. [3], the authors developed the USE (user-system-experience) model for describing gameplay. This system has been somewhat accurate at evaluating the success of an already developed video game; however, the system is difficult to use to evaluate a game under development or in the planning stages and is incapable of identifying flaws that might exist in an already developed game as it relies on a complete experience. The system focuses on the extremes of an *atomic level* gameplay experience and extrapolates from there to the broadest of possible outcomes such as "*fun*", which is defined as a reliance on *flow* [4]. While fun and flow are important possible player outcomes, they are not the only successful game outcomes. Many game sessions of Dark Souls end in the pure controller flinging frustration of failure, yet this is a successful game that players designate as "fun." Also, the extreme difficulty of horror games, such as the Resident Evil series, routinely use a film technique known as a *jump scare* to break the current pace of the game and to provoke a new, separate reaction from the player that breaks the previous flow.

Gregory [11] concludes that "*flow* is the key to video gaming's universal appeal" and he provides a new way to test whether or not a player is in the flow state through the *Experience Sampling Method* (ESM). However, everyone has a family member who doesn't play video games. Even within the group of people who enjoy gaming, not every game appeals to them. What good is an ESM test if it is used only to test good games on gamers? Sales figures can tell us whether or not a game is producing a successful experience with similar utility for the game designer.

What is needed is an approach that gives game designer's feedback on the interaction of individual game mechanics with each other, in regards to player experience. In Gee [10] the author comes far closer to making specific, actionable suggestions with relevance

towards game design. Gee presents thirteen motivating elements used within video games to produce engaging outcomes in their players. However, not all games use these all thirteen elements and the paper does little to discuss how these elements are used together to create good games.

Zegal et al. [25] provide a useful dissection of the game play experience. Time is subdivided into four common *Temporal Frames*: Real world, Gameworld, Coordination, and Fictive. These Frames are constantly interacting to produce the player's experience, and the intelligent manipulation of these Frames has been used to define many interesting games. What is lacking here is the fact that game decisions take place on many different timescales inside of what Zegal would define as the same frame. The player of PacMan must dodge ghosts while constantly aware of the waiting power pills that could change the ghosts into targets. However, that ability is limited and must be used sparingly. These are two different decision cycles that are constantly happening inside of the player, but both happen at different rates.

Chapter 4

Player Decision Cycle Analysis of Skyrim

The Elder Scrolls V: Skyrim has been a successful game platform since it's introduction on November 11, 2011. It still exhibits a vibrant, player maintained modding community to this current day, six years after its introduction. To create an engaging gameplay experience, it is important to study and understand games that have already established a successful reputation. For that reason, in this chapter we dissect the design and motivating elements of Skyrim in detail. Using this analysis we will illustrate the general framework of the Player Decision Cycle approach, and then use it as a framework to discuss the games of PacMan and Civilization.

4.1 The Action Genre in Skyrim

Skyrim is a fantasy themed video game consisting of three primary genre types, as illustrated in Figure 4.1. These three genre are RPG, Open world, and Action, shown in clockwise order starting at the top of the figure. These three genres form a multi-directional interaction, illustrated by the arrows in the figure, that guide the player's engagement with the game and the world of Skyrim at different times and paces. A player who spends time with this game will become familiar with all three of these elements, illustrated in Figure 4.1.

Player engagement in a video game must be managed as a rising and falling sequence of actions much like pacing in a film. The action genre, as detailed in Figure 4.2, makes up the most short term and fast paced of player engagement mechanisms in Skyrim. This is most apparent in Skyrim's combat design.

Detailing this fast paced interaction, the fastest part is how the player controls their damage output. The player must position their avatar and attack in such a way as to be able to apply damage to the enemy NPCs while minimizing personal risk. During the

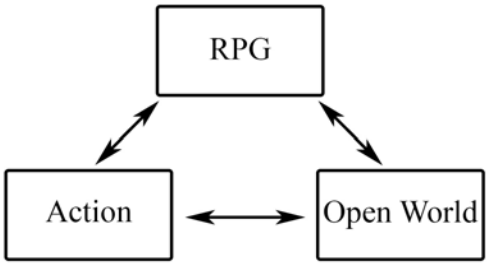


Figure 4.1: **Skyrim Genre Diagram**. This figure illustrates the three genres that interact to create the immersive game Skyrim.

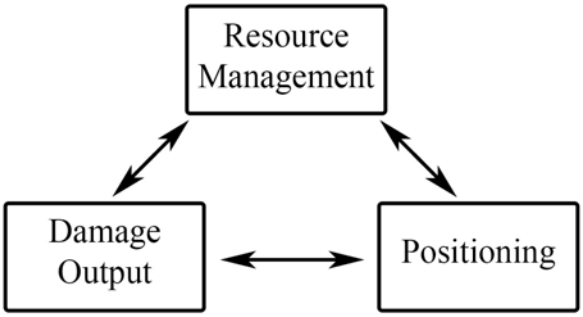


Figure 4.2: **Skyrim Action Detail**. This figure illustrates the interacting game elements that mediate player interactions with fast paced encounters in Skyrim.

course of this move and attack cycle, the player must also manage several assets to provide a slightly longer term tension building. In particular, they must judge the rate at which their health is depleting. They must spend mana to cast offensive spells while ensuring they have sufficient left to maintain healing and defensive spells. They must also manage the constantly regenerating Stamina bar to maximize their weapon damage output versus spending it on sprinting around the battlefield.

Finally, the player must manage the pace of their combats and the enemies they engage. For example, a combat can be avoided through the use of stealth game elements if the player is low on health potions or is suffering from battle fatigue; or simply, the player may be bored of fighting bandits and wishes to avoid combat to further explore the world. If engaged in combat, damage is applied to enemies in many different ways. Some enemies may be more easily damaged by specific types of attacks, so it is advantageous to apply that

type of damage to them preferentially. For example, when battling a dragon it is typically more expeditious to exploit ranged weapons such as bow and arrow rather than engaging the dragon directly with a sword and shield. It can also be advantageous to engage weaker enemies preferentially to remove them as a damage source for the remainder of the fight. For example, skeletons are typically easy to dispatch but may become a nuisance if not eliminated early on. This broader tactical thinking allows the player to engage the action elements of the game at somewhat more thoughtful pace. These three action elements work together to create a dynamic tension that alters moment to moment during combat encounters. Fast paced excitement is further leveraged by the dynamic nature of the game's NPCs. Many NPC enemies are designed to scale in difficulty relative to the level of the player, so that the challenge of combat will remain tense and exciting without falling too deeply into frustration or too easy to kill prey.

4.2 The RPG Genre in Skyrim

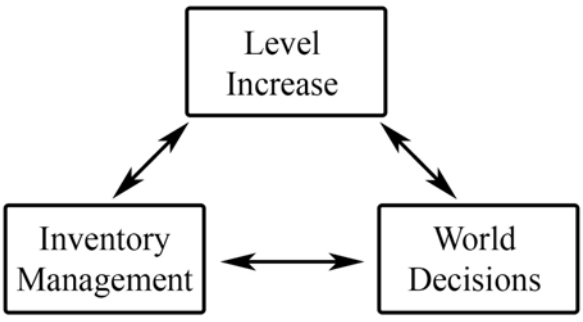


Figure 4.3: **Skyrim RPG Detail**. This figure illustrates the interacting game elements that mediate player interactions with their avatar and the narrative of the game world.

The RPG elements of Skyrim represent a middle ground in its pacing elements. With RPG elements the choices take place far more rarely than those during combat, but consequently have a far more pervasive effect. The RPG genre, as detailed in Figure 4.3, centers around granting the player control over their avatar's abilities, interactions with the narrative, and the items to be used in combat, shown in clockwise order starting at the top of the

figure.

A player may exchange their gear and weapon load at any time during the game. In the middle of combat, the player can take a break by simply opening up a menu and looking through their inventory. Necessarily, this breaks the game cycle of the action elements, but changing the weapons or spells the player is using has a direct effect on the result of the combat. To properly combat an enemy using a slow weapon, the player may equip a shield to try to stun them during the long build up to it's use. The player is encouraged to set up presets of weapons and spells, to speed the switch in combat.

The middle ground of the RPG elements is the management of future development of the player's character. As the player uses various interaction methods such as stealth, weapon use, and armor use, a skill associated with that use is trained and their ability with that specific skill is increased statistically. There are 18 different skills partitioned among three major interaction groups: Magic, Combat, and Stealth. Leveling up within one of these interaction groups rewards the player's perk points that can be spent on progressively larger increases as they gain more and more ability. The player is allowed to guide their player's long term development by choosing whether to cast a spell or swing a sword. This process provides a tension that acts against the short term use of skills purely for tactical advantage and rewards more deep specialization in specific skills.

The longest term thought process from the RPG elements is focused on player choice. When the game begins, the player is asked to design their character. They are allowed to choose their looks, race, and initial abilities. This choice will affect the game in a pervasive way throughout their entire play through. As the game begins, the player is tossed into the middle of a civil war that threatens to tear the province of Skyrim apart. Over the course of the game, the player must make choices that will support either side of this war, ultimately influencing the entire future of the region.

While the choices the player makes have little to do with the moment to moment game elements, it provides a sense of context to all of their actions. Leveraging the long term curiosity and providing purpose to the player serves to further immerse them in the world

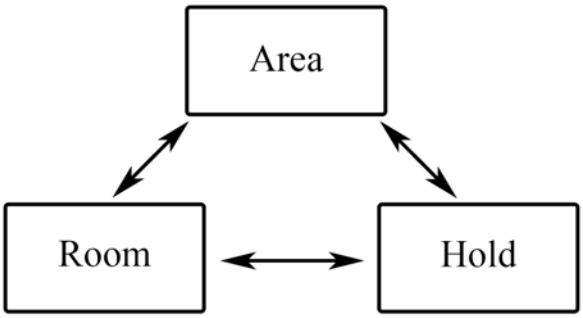


Figure 4.4: **Skyrim Open World Detail**. This figure illustrates how the level design of Skyrim mirrors the paced decision making process of the Action and RPG game design.

of Skyrim. It pushes them to look deeper into the world. As they become more immersed, they find a gigantic world that can engage them when other tasks begin to feel repetitive.

4.3 The Open World Genre in Skyrim

The Open World of Skyrim can be partitioned into three major level design areas, as illustrated in Figure 4.3. These level design units partition the map into pieces that the player can identify from a distance. This allows the player to make decisions about where to travel and have a reasonable assurance that the challenges and narratives found within will be related. Clockwise from the top of Figure 4.3 these level design groupings are Area, Hold, and Room.

The large open world map of Skyrim is partitioned into large zones that are referred to as *holds* in the game. These zones usually have a dominant biome that ties the area together thematically and a central city that serves as a hub that can direct a player to specific adventures within the area. A player can travel through these *holds* many times over the course of their time in Skyrim. The beautiful, complex landscapes beckon the player over the next horizon. This open, inviting feeling is at the core of the long term exploration mechanic in Skyrim. In direct opposition is the fact that enemy NPCs wait around every corner. This slows the player's progress across the world and adds to the feeling of distance that is in many ways an illusion. While the open world of Skyrim is 6.8 km by 5.4 km, this

area feels much larger because of the many obstacles and encounters in each area.

Each hold or zone has many smaller areas within in it. These areas subdivide *holds* into specific groups of encounters. In many cases, these groups of encounters are further linked by narratively designed quests. Generally, these quests guide the player through a specific series of obstacles and enemy NPCs while rewarding them with skills, items, and lore. These areas are further subdivided into rooms. A room is the basic component of Skyrim level design. In a room, individual encounters play out within limited confines. This design allows encounters to be sculpted for player ability and level. It also allows the designer to introduce linearity into the game for the sake of narrative.

Skyrim manages a complex nonlinear pacing system by allowing slow-paced player choice to direct them through a large scale world. Within this world, action elements serve to break up the more thoughtful exploration elements with high energy excitement. The balance between action and exploration is maintained by the RPG elements of the game, which provide a framework within which the player can construct a unique narrative for their journey through the game. This narrative drives the player ever deeper into the game and is at the core of the success of Skyrim in the long term. To understand this, the game-play experience of Skyrim must be described further. Unlike some game play experiences, Skyrim is designed to be enjoyed over a long period encompassing many play sessions. Estimates for the amount of scripted content estimate the typical game play experience at around 300 hours.

The scripted content of Skyrim is organized into many different quests which each have a linear progression from beginning to end. These quests are often further organized by not appearing until other quests have been completed. If a player is to experience a large amount of this content, they will need to come back to the game many times and complete many quests.

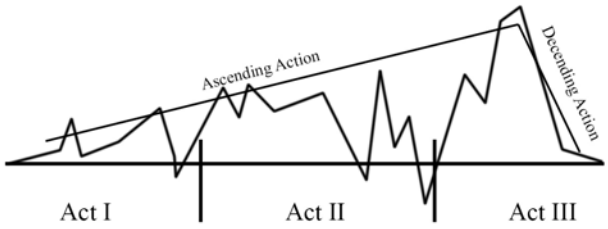


Figure 4.5: **The Three Act Structure**. This figure illustrates the way tension is often designed into linear narrative. Tension increases through the first two acts as idealized by the Ascending Action line, while the third act works as a climax and then provides resolution through Descending Action. The jagged line below illustrates the actual pace of the narrative. The ups and downs of this lines pacing provide constantly changing levels of tension and relief to maintain interest.

4.4 Timed Decisions and Tension

Drama is the key to the creation of memorable experiences. In the creation of movies, this is often modeled with the three act structure as detailed in 4.5. Act one is the introduction, act two provides increasing tension, and act three provides the climax and falling action.

This three act structure is mirrored in the play experience of Skyrim. However, instead of only relying on linear drama, the game uses intensity of action to provide pacing changes to control the player's interest in different moments. This is accomplished through the overlapping use of the various play cycles. Action provides the short term attraction of the game. This gives the player small rises and falls in the interest of any individual player's experience. However, over time, repetitive combat can feel flat and become repetitive. Context and the upward feeling of increasing tension is provided in the RPG mechanics of the game. The pull of character development, both mechanically and narratively, drive the rising action of increasing tension through scripted narratives and emergent player goals. Falling action is provided by the rewards of character leveling choice and the change of areas in response to player choice.

These two interacting systems, Action and RPG, provide the majority of the experience

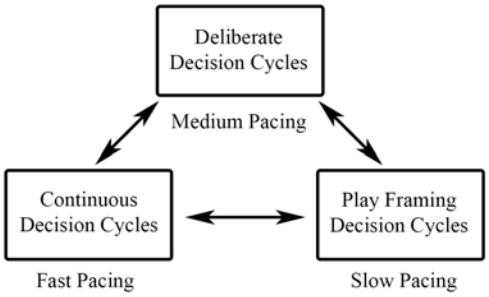


Figure 4.6: **Player Decision Cycle Framework**. This figure illustrates the general design framework that underlies the choice of what game elements are used to create dynamically changing tension in a single game session while luring the player back for multiple sessions. Continuous Decision Cycles are the fastest paced elements, Deliberate Decision Cycles have a medium pace, and Play Framing Decision Cycles have the slowest pace.

a player will have in a single setting. The promise of adventure provided by open world design is the draw that compels the player to return to a new play session. The organization of the world allows a player to come to a new session with the assurance that there will be a fresh adventure waiting. An old narrative could be continued or a new one started. That choice asks the player to connect to the adventure in a way that heightens the intensity and drama of the upcoming play session.

The push and pull of all of these competing systems during a play session could make for a confusing game experience. However, each system asks the player to make choices on different time frames. The action mechanics force a player to act and decide quickly. The RPG mechanics are slower and more deliberate, but the long term payoff of any decision gives each choice weight. The open world mechanics are so slow paced that the decisions made can shape the course of many play sessions and provide the connective tissue that binds the many adventures of the player into a cohesive whole.

The genres of RPG, Open World, and Action are used to create deliberate decision cycles, play framing decision cycles, and continuous decision cycles, as illustrated clockwise in Figure 4.6. The interaction of these decision cycles at different times and repeating rates form the basis of Player Decision Cycle (PDC) game design in Skyrim.

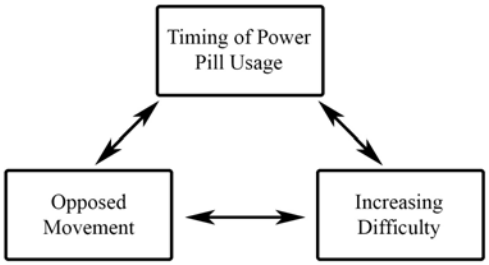


Figure 4.7: **Pac-Man PDC Diagram**. This figure illustrates the three interacting game elements that form the PDC framework of the Pac-Man game design. Clockwise from the top they are Timing of Power Pill Usage, Increasing Difficulty, and Opposed Movement.

4.5 The Player Decision Cycle Framework In Other Games

The creation of games is a very complex process. The management of player interest is at the core the creation of any game. The designer spends the currency of player interest every time they ask the player to do any task. It is important to use that interest to further the creation of more. This process of player investment and the return on that investment is at the heart of any gameplay experience. Sometimes, that return can be referred to as 'fun'. However, it is important to note that 'fun' is not the only positive outcome that can come from a gameplay experience.

Games exist as challenges. The early history of gaming is filled with games that ride a fine line between frustration and fun. However difficult or frustrating a game might have been, if the game succeeded at providing enough interest with the rest of it's mechanics, players would return. The frustration at failure in an interesting game drove the player to return to the world. In Figure 4.7 we detail the framework of the classic frustration based game of Pac-Man.

Movement provides the Continuous Decision Cycles of the game of Pac-Man. The player has clear goals in the form of dots to be eaten and points to be gained. Challenge comes in the form of enemies that try to move towards the player. Planning for the future provides the Deliberate Decision Cycles of the game, as the player plots the short term switch to offense provided by the power pills far in advance to use them to maximum effect.

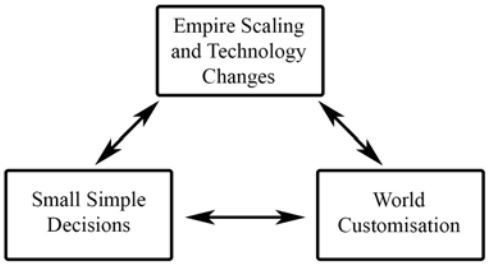


Figure 4.8: **Civilization PDC Diagram**. This figure illustrates the three interacting game elements that form the PDC framework of the Civilization game design. Clockwise from the top they are Empire Scaling and Technology Changes, World Customization, and Small Simple Decisions.

The Play Framing Decision Cycles of the game are provided by ever increasing difficulty. Pac-Man has no ending, only an eventual bug that ends the game experience during an integer overflow. Is this ending 'fun'? From a game design perspective, our framework argues that fun doesn't matter. What does matter is that the effect of this frustration is to draw players to return to the game and face the challenge of it's systems.

While Pac-Man is very different in scope and technology from Skyrim, the two games have the same primary focus: Draw the player in to experience the gameplay and allow the player to create an interesting experience from the tool kit of interacting systems. The game simply provides the framework that allows this experience to happen. A game that squanders the currency of attention is one that the player puts down quickly.

Not all games rely on the moment to moment decision making of action elements to provide the short term interest of it's game design. In Figure 4.8 we detail the framework of the slow paced game of Civilization. Continuous Decision Cycles in Civilization are based around making slow paced, seemingly simple decisions every turn. There is no time pressure to make these decisions, but the outcome of those choices is only felt in future turns. The player makes decisions, and then ends a turn to see the outcome of those decisions. Deliberate Decision Cycles are provided by the growth of the player's empire and the player's gradual increase of technologies over many turns. The Play Framing Decision Cycles of this game are created by the variability of the playspace and it's customization. What is

important in this example is to note that each of these game elements happens on different time scales with *respect to each other*. The Continuous Decision Cycles of Civilization are closer in pacing to that of the Deliberate Decision Cycles of Skyrim. This difference between Skyrim and Civilization 6 draws parallels to the difference between a drama and an action film. Both use similar structures to create interest, to very different results.

Between these two examples, and the more detailed analysis of Skyrim, we can see that interesting games require player decision making on different time scales. Player experiences like 'fun' and 'excitement' are outgrowths of game elements interacting to produce changing levels of tension over time. When one or more of these game elements conflict in decision timing and pacing in relation to other elements, the game experience will tend towards 'bad' or 'annoying.' As we explore the elements of game design in more detail, such as in the Skyrim RPG Detail of Figure 4.3, we see the same pattern of Continous Decision Cycles, Deliberate Decision Cycles, and Play Framing Decision Cycles. This causes the feeling of separation between the RPG game elements of Skyrim from the Action and Open World game elements, as in a sense, they are presented as separate games.

The flexibility of using the PDC Framework allows us to separate out elements of a game and discuss those elements in relation to the parts of play that share similar player interfaces and decision pacing. For example, we can ask if the Action game elements of Skyrim overlap with decision pacing of the other game elements within Skyrim. Does the Item Management pacing mesh well with the Damage Output, Resource Management, and Positioning Action game elements found in Figure 4.2? As the fastest paced part of the RPG game elements 4.3, this interaction could be a source of conflict that reduces player interest. As Bethesda has altered how Item Management, specifically weapon swapping, worked between Skyrim and their next project, Fallout 4, we can see that the developers of that game also identified this as a possibly problematic area of gameplay.

4.6 The Player Decision Cycle Framework Across Designs

The Player Decision Cycle Framework, seen in Figure 4.6, applies across video game designs. Within Skyrim, the genres of Action, RPG, and Open World are used to create the Player Decision Cycle Framework. Continuous Decision Cycles are provided by Action game elements as seen in Figure 4.2, Deliberate Decision Cycles are provided by the RPG elements seen in Figure 4.3, and the use of Open World game elements in Figure 4.3 provides Play Framing Decision Cycles. Skyrim attempts to balance these three cycles in it's game design. However, not all games share this balanced game design. In the case of games such as PacMan and Doom, the structure of the game design is focused on Continuous Decision Cycles. Civilization is a game design focused on Deliberate Decision Cycles. Game designs such as MineCraft and Dwarf Fortress are focused on Play Framing Decision Cycles. While these games may have different thematic designs, expected play experiences, and play pacing, they all use similar engagement strategies.

Chapter 5

The Player Decision Cycle Design of Dragon Mist and Radium Seas

In this chapter, we provide an overview of Dragon Mist and Radium Seas, two games with very different themes that draw upon the same game-flow engagement mechanics described in Chapter 4, and illustrated in the Elder Scrolls V: Skyrim. In the next section we describe Dragon Mist, and in Section 5.2 we describe Radium Seas.

5.1 Dragon Mist

> Work consists of whatever a body is obliged to do. Play consists of whatever a body is not obliged to do.
>
> *Mark Twain*

Dragon Mist is a Skyrim mod[1] that we designed and implemented using the general Player Decision Cycle Framework in Figure 4.6. The intention of this project is to leverage the game elements proven to work within Skyrim to drive an outcome that balances entertainment with an increase in player knowledge of genetics. This project was developed through the efforts of Dr. Meihua Qian, Dr. Samuel Sparace, Dr. Brian Malloy, and Rebecca Clark. To this end, the analysis of the Player Decision Cycles within Skyrim was used to help identify areas within the design of Skyrim that would allow for the addition of real-world education content without fundamentally damaging the Skyrim play experience.

In Figure 4.6 we see the three major decision making cycles present in Skyrim. Continuous Decision Cycles, who's Skyrim game elements are detailed in Figure 4.2, have been used in the past in games such as *Number Munchers* to provide learning experiences [2;

[1] The construction of *mod*s for Skyrim was described in Chapter 2

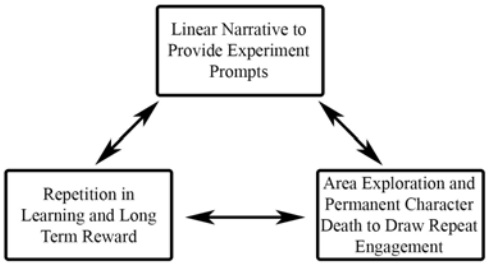


Figure 5.1: **Dragon Mist PDC Diagram**. This figure illustrates the changes to the the three interacting Player Decision Cycles in Skyrim when read clockwise starting at the top.

21]. This game rewards players who can answer questions quickly by providing them with points. The questions, though, are presented far more slowly, in the form of text at the beginning of a stage. This presentation is similar to the Deliberate Decision Cycle elements seen in Skyrim's RPG system, detailed in Figure 4.3. In this manner, we reinforce a complex learned concept with fast paced, repetitive actions found in Continous Decision Cycles. The final part of the framework needed is the Play Framing Decision Cycle, used in Skyrim as Open World game elements detailed in Figure 4.3. How do we allow the player choice in when to engage with the learning experience?

In Dragon Mist, we created additions to each of the three major cycles present in the game of Skyrim as detailed in Figure 5.1. The RPG cycle was used as the primary vector for the genetic information being transmitted to the player. This took the form of a new linear narrative or quest. This quest uses dialogue and books to introduce the player to the dragon workbench. At this workbench, the player is encouraged by dialogue to create a dragon companion from samples in the surrounding area. This dialogue and a book found near by also serves to explain the underlying genetic processes being demonstrated during the experiment.

The Action cycle is primarily to create dynamic tension in the same manner as Skyrim. However, once the player reaches the workbench it is used to help reinforce the learning goals in two ways. First, using the workbench to create a dragon has a chance of failing to create a docile companion. When this happens, an aggressive dragon is instead created,

forcing a conflict with the player. This serves as a pacing change and an opportunity to have a learning dialogue in the narrative questline. Secondly, if the player does create a docile dragon it's primary use in game is as a combat companion. It provides the player with a change to the combat environment outside of the local confines of the mod.

The Open World cycle is used in two ways as well. First, the mod provides a new area of exploration to be discovered by the player. Secondly, the dragon companions created during the experiments are able to die permanently. This means the player is forced to reengage with the mod any time they wish to replace their companion. This allows the mod to interact with the Play Framing Decision Cycles of Skyrim even after the player has finished the linear narrative of the mod.

Skyrim expertly creates a visually compelling world that the player wants to explore. Dialogue, books, and history litter the environment. By replacing fictional world building with our learning goals we provide an environment where the player's natural interest drives them to learning. Combat is used to drive interest in the story, while occasionally being used to reinforce the learning goals of the mod and to break up the slow pacing of experimenting with dragon creation. We try to drive the player to return by introducing a lack of permanence in their dragon creation. This allows the player to engage with the mod when they choose to, but positively rewards them when they decide to return.

5.2 Radium Seas

Creating a new video game is complicated, time consuming, and fraught with the risk of lacking engagement and thereby not connecting properly with the audience. We wish to use what we have learned from Skyrim to help guide us toward making a new, engaging, entertainment experience.

Producing a new experience means starting with a new theme. The narrative of Radium Seas is based around the player controlling an independent pirate ship after a nuclear war has devastated America. This thematic change demands many changes in the way the basic

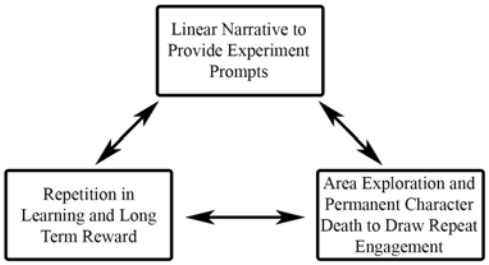


Figure 5.2: **Radium Seas PDC Diagram**. This figure illustrates the major game cycles introduced to create the game of Radium Seas, as mapped on to the standard framework from Figure 4.6.

Skyrim game elements of Action, RPG, and Open World, detailed in Figure 4.6, are used. Radium Seas uses a simplified version of the same base structure from Skyrim, detailed in Figure 5.2.

The Continuous Decision Cycles in Radium Seas are provided by a combat system that emphasizes positioning and fire timing. The player's pirate ship avatar maneuvers differently depending on the wind speed and direction. To maximize damage to enemy ships, the player must turn their ship to have as many guns facing their target as possible. The guns can only fire occasionally, so the player must make the turn towards their target at the right time to be able to fire. The health of the player must be managed while this is occurring.

Deliberate Decision Cycles are provided by the characterization of the ship as the player's avatar and narrative elements attempt to engage the player in the life of a ship captain after the apocalypse. Points earned in combat are used to heal the ship and to finish the narrative.

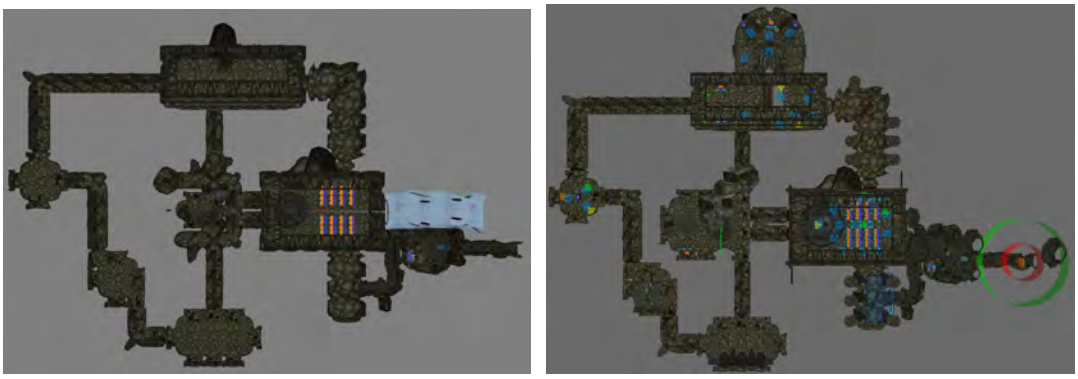
Play Framing Decision Cycles are provided by the world exploration elements which allow the player the ability to see encounters from far enough away to choose how to engage them. Is a battle worth fighting, or is a dangerous area worth exploring are the questions that guide player movement.

Chapter 6

The Implementaton of Dragon Mist

Dragon Mist is a new quest line, implemented as a mod, that can be played within the Elder Scrolls V: Skyrim. The focus of Dragon Mist is dragons, also know as Dovah. According to the lore of Skyrim, dragons emanated from the continent of Akavir and though they appear beastlike, they are actually intelligent creatures capable of speech and written language. Their language includes the ability to cast magical shouts known as Thu'um or *dragon shout*. Dragons were once widespread in Skyrim and all of Tamriel but they were hunted and killed by powerful warriors in Tamriel until they were almost extinct. However, they began to reappear with the arrival of the most powerful dragon of all, Alduin, also known as the "World Eater." The main quest in the game Skyrim involves the player defending Skyrim against Alduin, culminating in an epic battle pitting the player and some ancient Nords against Alduin. Although Alduin is considered malevolent, not all dragons in Skyrim are evil and some dragons, most notably Paarthanax, actually assist the player in achieving various goals.

One of the goals of the Dragon Mist mod is to incorporate instruction in genetics into a video game. Thus, a problem with the game design of Dragon Mist is creating an interesting way of incorporating genetics through dragon breeding that fits within the existing lore of Skyrim. Dragons, in the game of Skyrim, are sexless, immortal creatures who were created by a god at the beginning of time. This challenges the central principles of sexual reproduction with genetic combination. To combat this problem, we developed a story involving a magic user from the ancient past of Skyrim, thereby creating the infrastructure necessary to allow the breeding of creatures who would not normally breed in the game. The narrative developed through the magic user allows us to utilize real world genetics while not straying far from the Skyrim game world and its existing lore.



(a) Empty Dungeon (b) Final Dungeon Layout

Figure 6.1: These figures show the iterative construction of the Dragon Mist dungeon from initial concept, shown in Figure 6.1a, to final presentation shown in Figure 6.1b.

6.1 Dragon Mist Dungeon

The magical breeding narrative is represented in the overall theme of the area. The map design takes the form of an abandoned temple dedicated to the dragons where the now dead magic user, or dragon priest, is working on a novel method to resurrect his dead dragon overlords. The temple is populated with undead, called *drauger*, who are followers of the dragon priest and become obstacles in the path of the player. A final battle between the player and the high priest represents the climactic fight in the mod.

Figure 6.1 illustrates the construction of the play space from first concept to final presentation. Figure 6.1a shows the initial dungeon layout as an empty space that was used for the testing of more complicated scripting without the clutter involved in a full dungeon layout. For example, we did not have to kill an NPC to further test the dungeon layout. Figure 6.1b illustrates the final look of the dungeon from above with all of the interaction points, patrol points, and Creation Kit interface elements needed to give the dungeon a completed look and feel.

To support the idea of finding a lost temple, the zone begins in a dark cave, illustrated in Figure 6.2, built using game artifacts found in the creation kit. The low embers of a fire reveal the resting area of a hunter. A bow and tanning rack add to the notion that this area

Figure 6.2: **Main Entrance**. This picture shows the entrance to the hunter's cave, which is the beginning of Dragon Mist.

has been used recently.

Deeper into the cave, the path twists and leads upwards. Through a broken masonry wall the player finds a new area, illustrated in Figure 6.3, with the corpse of the hunter whose camp was seen earlier. As the player begins to examine the corpse of the hunter, the drauger guards of the area attack. This ambush, show in Figure 6.4 was built using a trigger volume found at the entrance to the room, a standard technique used to build traps in the game of Skyrim.

Through a set of double doors the player finds the main hall of the temple, as show in Figure 6.5. An acolyte of the dragons goes through the motions of preaching to the husks of worshipers. Each of the drauger follows a series of waypoints within this room. The priest stays on the dais, while the worshipers move among the pews, as seen in Figure 6.6.

Behind the dais is a damaged series of rooms. In Figure 6.7 we see a priest's office with a writing desk and dresser, adding to the world of the mod while hinting at the tragedy that befell this temple. The most direct path to the objective is blocked by a collapsed ceiling, as seen in Figure 6.8.

The in the next chamber, the way forward is hidden. To find it, the player must pull a lever at the base of a flame thrower trap, as shown in Figure 6.9. This trap is inspired by a

Figure 6.3: **Cave to Temple Transition**. This picture shows the entrance to the main temple area, where the cave gives way to an ancient temple.

trap in Indiana Jones and the Last Crusade, in which the explorer must kneel before God or be beheaded. Here, the room fills with fire, as seen in Figure 6.10, and only a person who approaches on bended knee can escape unscathed.

Beyond the secret door there is a room in which giant rats, called Skeevers, have invaded the temple and devoured the drauger guards as seen in Figure 6.11. The Skeevers serve as a simple obstacle in a challenging dungeon.

Farther ahead, Drauger Acolytes summon dark magics in the depths of the temple as seen in Figure 6.12.

The final hall serves as the setting of a climactic battle to end the combat portion of Dragon Mist. In Figure 6.13 Drauger guards stand at the doorway while the head priest wanders his library and research area.

The ritual chamber serves as a place to house the magical machinery for the breeding of dragons. A customized workbench, shown in Figure 6.14, allows access to the dragon breeding menu. From this menu the player can select dragon samples to use in creating new dragons.

Figure 6.4: **Ambush**. This picture shows drauger enemies ambushing the player.

6.2 Dragon Mist and The Creation Kit

The process of creating new dragons through genetic combination requires the development of a way to store genetic data for the new dragons and to allow the player to manipulate that data. The implementation of the Papyrus scripting language permitted the development of a system that permits the organization and storage of genetic data, found on objects in the world, within a quest different from the one which stored the dialogue to keep the variables from being reset as the player moved throughout the world. This storage technique required that the genetic information storage had to be handled in a separate script from the player interaction script. Once players find various genetic samples in the world, the main script choreographs the player's ability to manipulate those samples, now stored in the main genetic repository.

As an overview, the structure used to store this information was implemented with several lists of Boolean, as well as a few variables to handle how to look at the Boolean. The passive/aggressive trait was stored, per sample, as two Boolean with false representing the aggressive trait and true standing for the recessive trait of passive. In Figure 6.15, we see how the menu structure allows for the inspection of samples, the charging of the two genetic holders called the Stone and the Essence, and the ability to inspect the Stone and Essence

Figure 6.5: **Temple Main Hall**. This picture shows the main hall of the temple from the doorway.

for the traits stored inside.

A main list stores all of the player's retrieved genetic data. Two integers inform the script how to look at this list: one to tell how many samples were stored overall and another to say how many Boolean were being stored per sample. The selected samples are held in the Stone and the Essence until they are asked to be combined by selection in the menu. In Figure 6.16 we see the genes held in the Stone and Essence being combined randomly to create a new dragon. If the dragon shows both recessive traits, referred to as the Boolean True in the script, the dragon picked will not be aggressive.

To increase player engagement in the mod it is important to provide the player with a reward for completing the complicated task of creating a new dragon. This reward is the creation of a new dragon with the potential to be a companion for the player, assisting him in the tasks that he might wish to perform. The creation kit actor information is displayed in Figure 6.17. Creating a new dragon companion for the player required the creation of an actor with an attached script. The script allowed the dragon to join the player as a companion and allowed the dragon to use the companion interaction mechanisms built into Skyrim.

Using the companion actor as a base, we created an aggressive version of this dragon

Figure 6.6: **Temple from the Dias**. This picture shows the main hall of the temple from the dais.

as seen in Figure 6.18. If the player creates an aggressive dragon, which is likely given that aggression is the dominant trait therefore appears seventy five percent of the time, the ensuing battle serves to underline the educational point while providing a fast paced break from the slower paced learning.

With these systems in place, we created another quest, this one designed to link the various elements of the mod together. The primary interaction point of this quest is an NPC named Jo'Tsrhni Bhusari as seen in Figure 6.19. He is a Khajiit Mage who has found indications of a lost temple in the area with interesting magic to be explored.

Bhusari is designed to serve as an informational assistant to the player by providing background, history, and learning supplements. This companion reacts to the player's passage through the level, helps them fight, and directs them towards various goals inside the level such as the hunter's body in Figure 6.3. The quest begins in the first town a player is likely to journey to in the game of Skyrim, Riverwood. Bhusari can be found in the inn, *Sleeping Giant*. He has been recruiting a local hunter to help him find the Temple; however, the hunter has not returned. Bhusari now plans on following up on that hunter's lead with the help of the player. This serves as a launching point for the major events that happen 6.1.

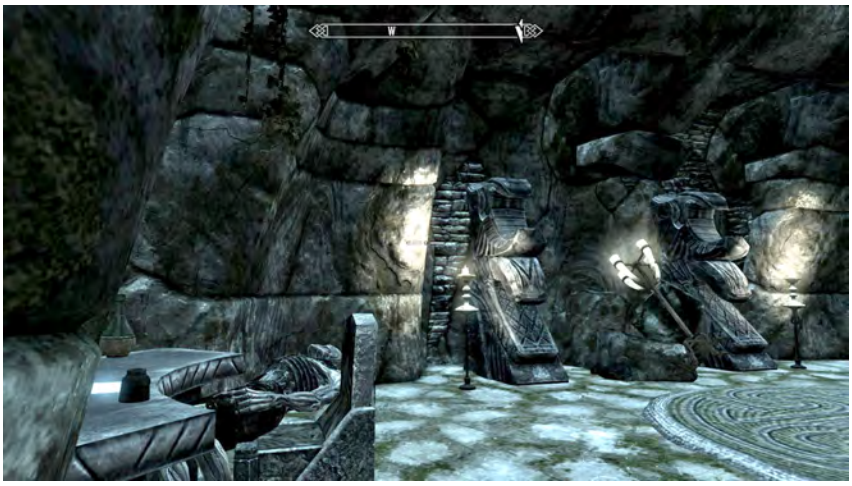


Figure 6.7: **Priest's Office**. This picture shows the priest's office from the entrance.

Figure 6.8: **Collapsed Tunnel**. This picture shows the blocked path to the final fight.

Figure 6.9: **Dragon God Trap**. This picture shows the dragon trap before activation.

Figure 6.10: **Activated Dragon God Trap**. This picture shows the dragon trap when activated.

Figure 6.11: **Skeever's Dinner**. This picture shows the Skeevers gathered around the remains of devoured drauger.

Figure 6.12: **Drauger Acolytes**. This picture shows acolytes around a magical flame.

Figure 6.13: **Temple Library**. This picture shows guards looking on as the head priest continues his work beyond death.

Figure 6.14: **Ritual Chamber**. This picture shows the ritual room where dragon's are created with a customized alchemy table at the center.

```
If akActionRef == Game.GetPlayer()
    int bMenu = menu.show()
    if bMenu == 1
        ; inspect samples
        int button = checkSamples1.show()
        int firstGene = button * GeneData.GenesPerSample
        int maxButton = GeneData.ActiveSamples
        ; if (button != 9 && button < maxButton)
            ;Debug.MessageBox("Sample " + (button + 1) + " shows " + GeneData.Genetics[firstGene] + " and " + GeneData.Genetics[firstGene + 1] + " for aggression")
        ; else
            ; Debug.MessageBox("Quiting Station")
        ; endif

    elseif (bMenu == 2 && ! GeneData.hasLStone)
        ; charge stone
        int button = checkSamples1.show()
        int firstGene = button * GeneData.GenesPerSample
        int maxButton = GeneData.ActiveSamples
        int randomChoice = Utility.RandomInt(0, 1)

        GeneData.EggGenes[0] = GeneData.Genetics[firstGene + randomChoice]

        GeneData.hasLStone = True
        Debug.MessageBox("Stone Charged, Quiting Station")

    elseif (bMenu == 3 && GeneData.hasLEssence != TRUE)
        ; charge essence
        int button = checkSamples1.show()
        int firstGene = button * GeneData.GenesPerSample
        int maxButton = GeneData.ActiveSamples
        int randomChoice = Utility.RandomInt(0, 1)

        GeneData.SpermGenes[0] = GeneData.Genetics[firstGene + randomChoice]

        GeneData.hasLEssence = True

        Debug.MessageBox("Essence Charged, Quiting Station")
    elseif (bMenu == 4 && GeneData.hasLStone == True && GeneData.hasLEssence == True)
        ; inspect living stone and egg
        Debug.MessageBox("Stone shows " + GeneData.EggGenes[0] + " and Essence shows" + GeneData.SpermGenes[0] + " for passive")
```

Figure 6.15: **Dragon Sample Controls**. This picture shows the script that controls the inspection and manipulation of dragon samples.

```
    elseif (bMenu == 5 && GeneData.hasLStone == True && GeneData.hasLEssence == True)
        ; create new dragon
        if MainQuest.MadeDragon == False
            MainQuest.MadeDragon == True
            DM_AbandonedTemple.SetObjectiveDisplayed(50)
            DM_AbandonedTemple.SetStage(50)
        endif

        reset = True

        GeneData.newDragonGenes[0] = GeneData.EggGenes[0]
        GeneData.newDragonGenes[1] = GeneData.SpermGenes[0]
        if (GeneData.newDragonGenes[0] == True && GeneData.newDragonGenes[1] == True)
;           Debug.MessageBox("New Dragon should be passive")
            pickDragon(0)
            SpawnedDragon = GeneData.DragonSpawnPoint.PlaceAtMe(DragonToSpawn)
            SpawnedDragon.MoveTo(GeneData.DragonTeleportPoint)
;           NiceDragon.Enable()
        else
;           Debug.MessageBox("New Dragon shows " + GeneData.newDragonGenes[0] + " and " + GeneData.newDragonGenes[1] + ".  New dragon should be aggressive")
            pickDragon(1)
            SpawnedDragon = GeneData.DragonSpawnPoint.PlaceAtMe(DragonToSpawn)
            SpawnedDragon.MoveTo(GeneData.DragonTeleportPoint)
            MeanDragon.Enable()
        endif
    else
        Debug.MessageBox("Quiting Station, stone/essence" + GeneData.hasLStone + " " + GeneData.hasLEssence)
    endif

EndIf


EndEvent

function pickDragon(int dragonRef)
    Debug.MessageBox ("Dragon picked" + dragonRef)
    if dragonRef == 0
        DragonToSpawn = GeneData.DragonSpawnList.GetAt(0)
    elseif dragonRef == 1
        DragonToSpawn = GeneData.DragonSpawnList.GetAt(1)
    else
        Debug.MessageBox ("Didn't Pick a Dragon From The List")
    ENDIF

EndFunction
```

Figure 6.16: **Dragon Creation Script**. This picture shows the script that controls the creation of new dragons with collected samples.

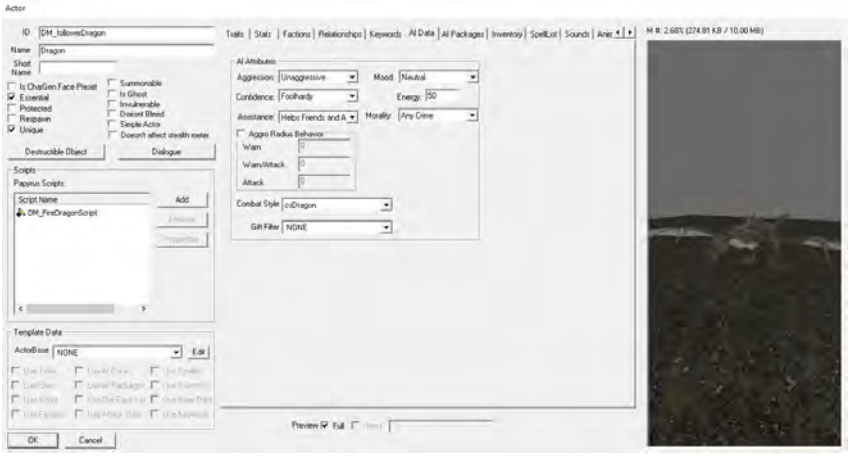


Figure 6.17: **Companion Dragon Actor**. This picture shows the Creation Kit actor settings for the created helpful dragon companion.

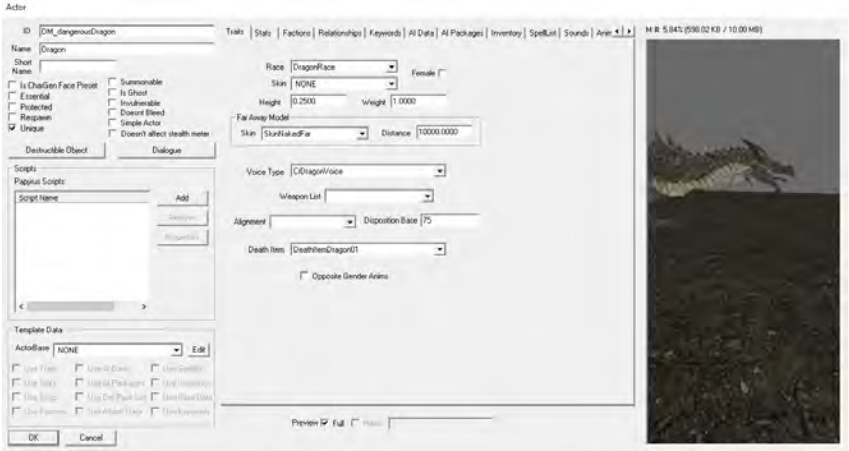


Figure 6.18: **Aggressive Dragon Actor**. This picture shows the Creation Kit actor settings for the created dangerous dragon enemy.

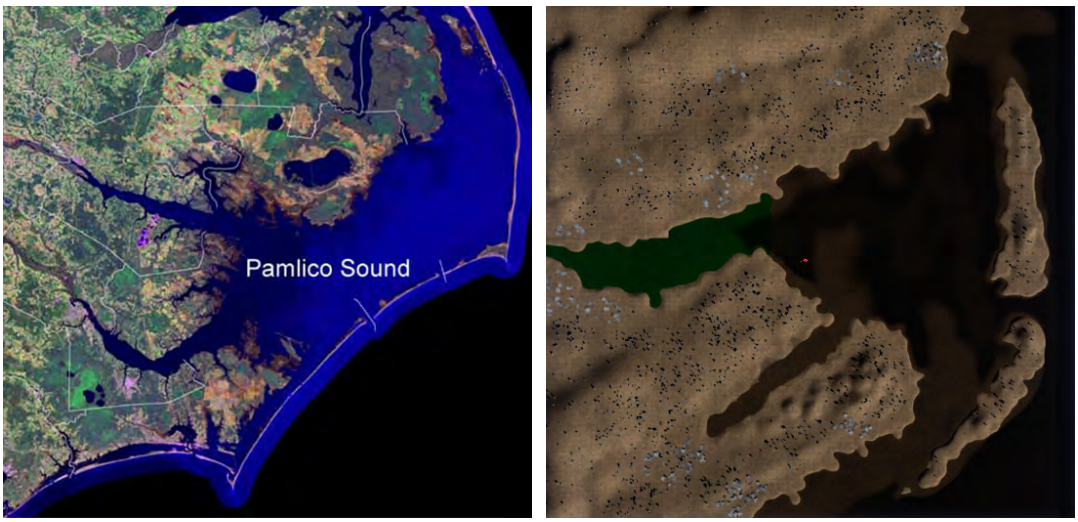
Figure 6.19: **Jo'Tsrhni Bhusari**. This picture shows the NPC mage who helps lead the player through the mod Dragon Mist.

Chapter 7

The Implementaton of Radium Seas

In this chapter we describe our implementation of Radium Seas, which we developed by adapting the themes found in the Fallout series, developed by Bethesda, and the Pirates series, developed by Sid Meiers [20; 22].

Our goal in constructing Radium Seas is to build a new game that combines the gaming concepts found in Fallout and The Pirates. Thematically, the lawless worlds of Fallout and the age of piracy align very well. Since Fallout is rooted in American lore, the design focuses on historical areas of piracy in America. The famous pirate Blackbeard was killed off of the coast of North Carolina in the Pamlico Sound in 1718, as seen in Figure 7.1a. This naturally self-contained sea area became the inspiration for the Radium Seas map designs, as shown in Figure 7.1b.



(a) Satalite Map of the Pamlico Sound (b) Radium Seas Play Space

Figure 7.1: These figures show a comparison between the Pamlico Sound, shown in Figure 7.1a, to the Radium Seas game map shown in Figure 7.1b.

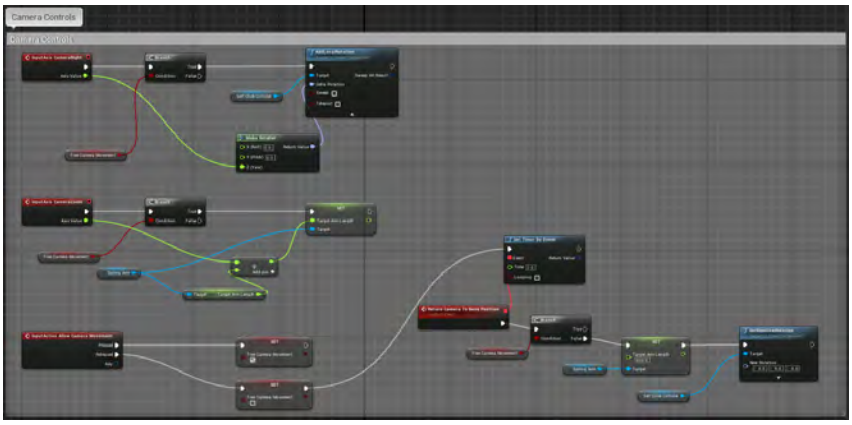


Figure 7.2: **Controls Blueprint**. This picture shows the blueprint that handles player inputs for controlling the camera.

7.1 The Player Controller

A game requiring piracy naturally revolves around ships. To insure this focus, the player controller of the game is based around a refitted fishing trawler. The game is controlled from a third person perspective similar to to the over the shoulder view of Fallout 4's third person camera and the third person isometric view of Sid Meier's Pirates. The camera uses a spring arm attached to the base actor. This allows the camera to follow behind the player's ship smoothly, while interacting with geometry in the world. Controls for the camera are mapped to pressing the 'control' key and rolling the middle mouse wheel. After the player releases the 'control' key, the camera snaps back to its original position. The implementation details for this camera input are illustrated by the blueprint shown in Figure 7.2.

7.1.1 Player Movement

Movement of the player character is handled with a different blueprint. Axial controls, which handle left and right rotation, are mapped to the 'A' and 'D' keys. Using these keys issues an output event in the blueprint, which turns the ship according to a variable assigned to the controller known as Turn Rate. The 'W' and 'S' keys change the level of the sails. The 'E' key activates engines, which provide the player with a continuous movement power, regardless of the wind, for a short period of time. These interactions are detailed in 7.3.

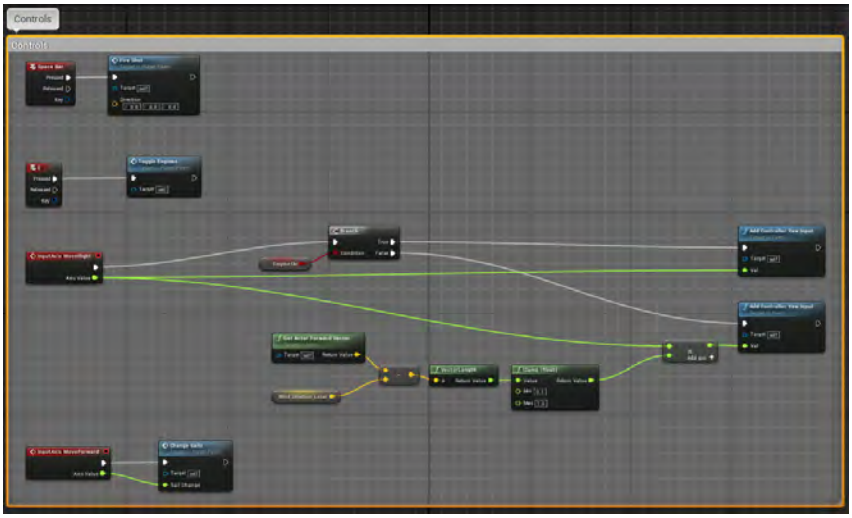


Figure 7.3: **Controls Blueprint**. This picture shows the blueprint that handles player inputs.

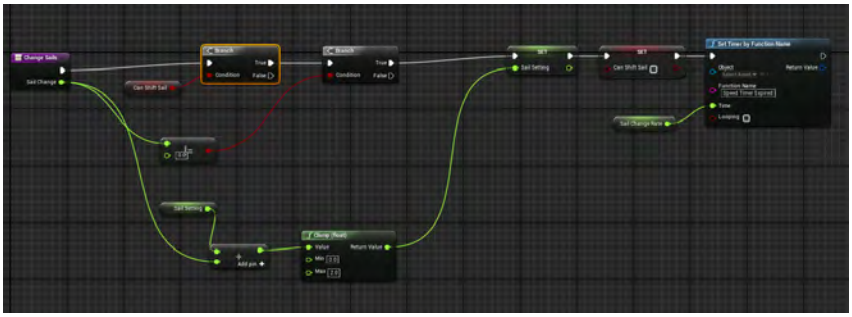


Figure 7.4: **Change Sails Function**. This picture shows the function that handles the player's ability to change sail settings.

Movement of the player controller is modified by wind. This effects the player controller in two main ways: Maximum speed modified by sail setting, and turn rate modified by wind direction. There are three sail settings: *reefed*, *half*, and *full*. When the sails are reefed, the ship has no interaction with the wind and can only turn. When the sails are at half or full, the ship moves forward in relation to the wind speed and direction. This is handled by the function *Change Sails*, detailed in Figure 7.4.

The turn rate variable is modified by current direction of the wind, called *Wind Direction Local*, with turning becoming more difficult as the closer the player's heading is towards the vector of the wind. To insure the ability for the player to have some control regardless of wind direction, this turning speed reduction is clamped to a minimum of ten percent of

Figure 7.5: **Close Up of Axial Controls**. This picture shows the Blueprint that handles how the player's turning inputs are modified by the wind direction.

Figure 7.6: **Update Phase of Movement**. This picture shows the Blueprint that handles how the player controller's movement modified.

the ship's maximum turn rate.

Movement of the player controller is handled in three main steps: Update, Create Movement, and Test For Collision. The Update Section, as seen in Figure 7.6, shows how the game is constantly checking the functions Update Wind, detailed in Figure 7.7, and Check Speed, detailed in Figure 7.8, against current player settings.

During the Create Movement Phase, detailed in Figure 7.9, the Blueprint modifies the player controller's speed by how much time has passed with the Delta Time variable. This is then combined with the current forward vector of the player controller. If that vector's length is larger than zero, the Blueprint moves the player controller along that vector with the AddActorWorldOffset node. That node checks the area in front of it with a sweep, which is critical in the next phase.

During the Test phase of movement, as detailed in Figure 7.10, the player controller tests the area that it is attempting to move into for blockages. The sweep results from the previous phase are analyzed and if there is a Blocking Hit, the controller attempts to slide

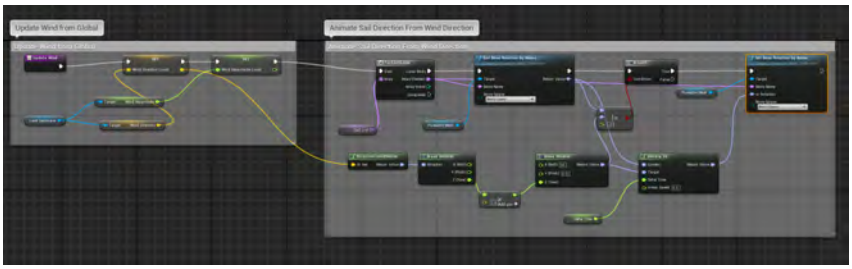


Figure 7.7: **Update Wind Function**. This picture shows the Blueprint that handles how the player controller's checks for the current wind direction and then modifies the sail's of the ship to turn with it.

along the outside of it.

7.1.2 Weapons

Weapon use in Radium Seas is handled in three steps: The player selects a target by clicking on in using the mouse, the player chooses to fire by pressing the spacebar, and damage is resolved when projectiles collide with a target.

Targeting is achieved using a trace function bound to the left click of the mouse, as detailed in Figure 7.11. If the player has clicked on a valid target, the trace function feeds the resulting actor to the ship into a variable for later use.

The target variable is fed into the Aim Guns function, detailed in Figure 7.12, which iterates through the each weapon position on the ship and turns them to face the target if it is within a set number of degrees. If the gun cannot turn to the target, it turns as close as it can will not fire. If the gun can see the target, it is allowed to fire on command.

When the player presses the spacebar with a valid target selected, the list of available guns is checked. At each available gun, a projectile is spawned from the tip of the active gun which moves towards the target.

If a projectile hits a target it reduces a hit points variable, HP. If HP is reduced to zero, the target is destroyed. Each valid target, including the player controller, has a copy of this Take Damage Blueprint, detailed in Figure 7.14.

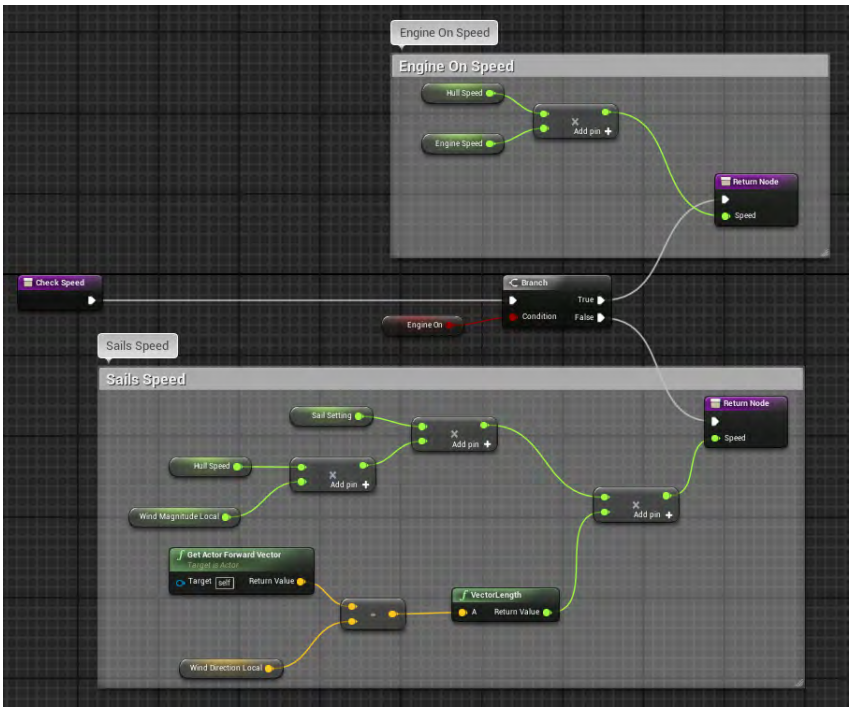


Figure 7.8: **Check Speed Function**. This picture shows the Blueprint that handles how the player controller's speed is modified by engines, sail settings, and wind speed.

7.2 The Level Database Actor

The wind direction, as well as other information important to multiple actors in the game, is stored outside of any specific actor, such as a ship or town, in the game. The actor that creates and stores this information is known as the Level Database Actor. All ships in the game have a reference to this actor to allow themselves to find the current wind speed and direction. Wind speed and direction are generated randomly in the level database actor at a period determined by the variable Wind Change Rate, as detailed in Figure 7.15.

7.3 Enemies

Opponents in the game are created and spawned using the player controller as a base. The wind interaction and weapon turning mechanics are carried over from the player controller. However, to give the AI ability to use these functions, new techniques are needed. This

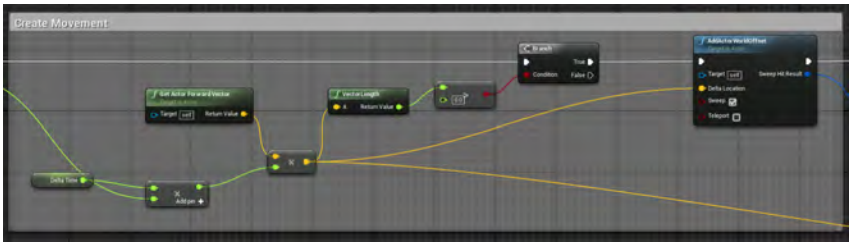


Figure 7.9: **Create Movement Phase of Movement**. This picture shows the Blueprint that handles how the player controller's direction and speed are used to create movement.

Figure 7.10: **Test Phase of Movement**. This picture shows the Blueprint that handles how the player controller's movement modified.

is handled in the Behavior Tree. The AI stores two locations constantly, Home and Destination, and attempts to move towards another location called Target To Follow. Initially, Target To Follow is Destination. If the actor reaches that point, it waits and then switches Target To Follow to Home. If the ship encounters an enemy, or if its Destination is an enemy, the actor attempts to fight it. This Behavior Tree is detailed in Figure 7.16.

Movement is controlled in a new way from the player controller. If the actor is a long distance from it's target, the actor will attempt to maximize its speed in a sailing process known as tacking. Tacking turns the ship from a direct course towards it's target towards either the left or right as show in Figure 7.17.

Tacking movement is created by the actor as it moves towards its destination, as detailed in Figure 7.18. Here, during the Tack or Move To Target phase, the actor checks how far it is from it's destination. If it is close, it will move directly towards it's target. If the actor is

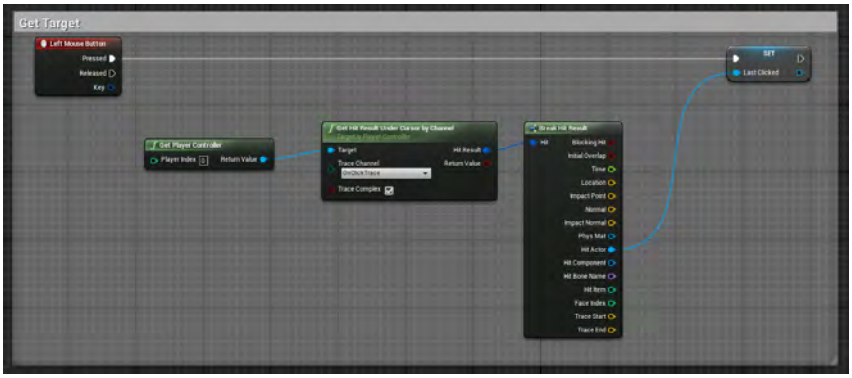


Figure 7.11: **Player Targeting**. This picture shows the Blueprint that handles how the player's left click stores a target for later use in the Last Clicked variable.

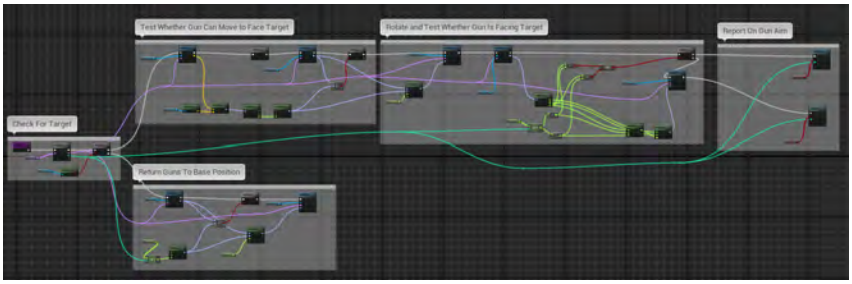


Figure 7.12: **Weapon Targeting**. This picture shows the Blueprint that handles how the player's left click stores a target for later use in the Last Clicked variable.

far away, will either move towards it's target or the tack location. The ship moves to tack every few seconds, returns to a direct course, and then tacks in the opposite direction, and so on. To make sure these tack locations don't cause the ship to run aground, the location is created by invisible objects attached to the main collision area of the ship by Spring Boom objects. If the length of the Spring Booms has changed, the ship will not attempt to tack towards it.

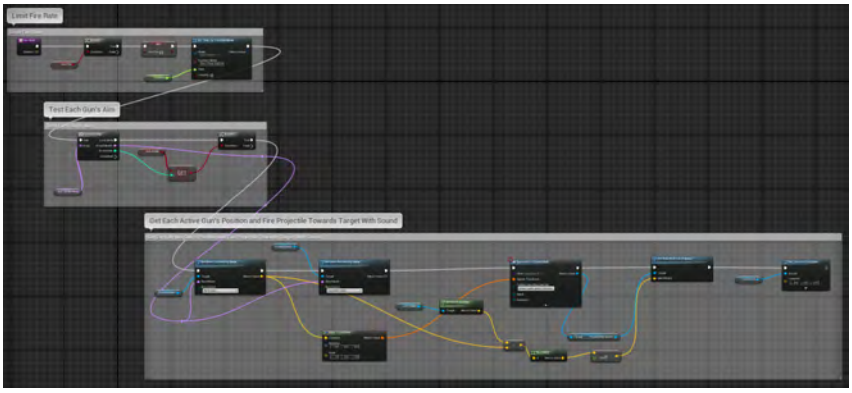


Figure 7.13: **Firing**. This picture shows the Blueprint that handles how the player's press of the spacebar fires every gun that is turned to face the target.

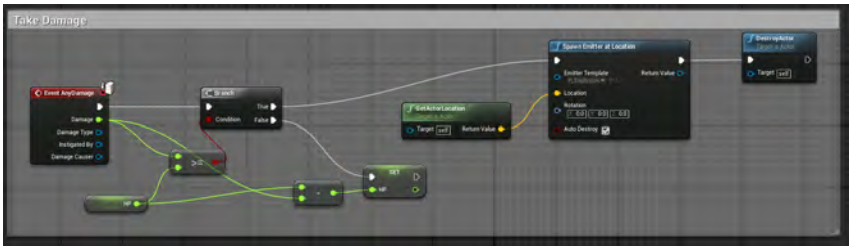


Figure 7.14: **Take Damage**. This picture shows the Blueprint that handles how the all actors, including the player controller, handle interaction with damaging projectiles.

Figure 7.15: **Change Wind Function**. This picture shows the Blueprint that creates new wind vectors and and magnitudes at a period determined by the Wind Change Rate.

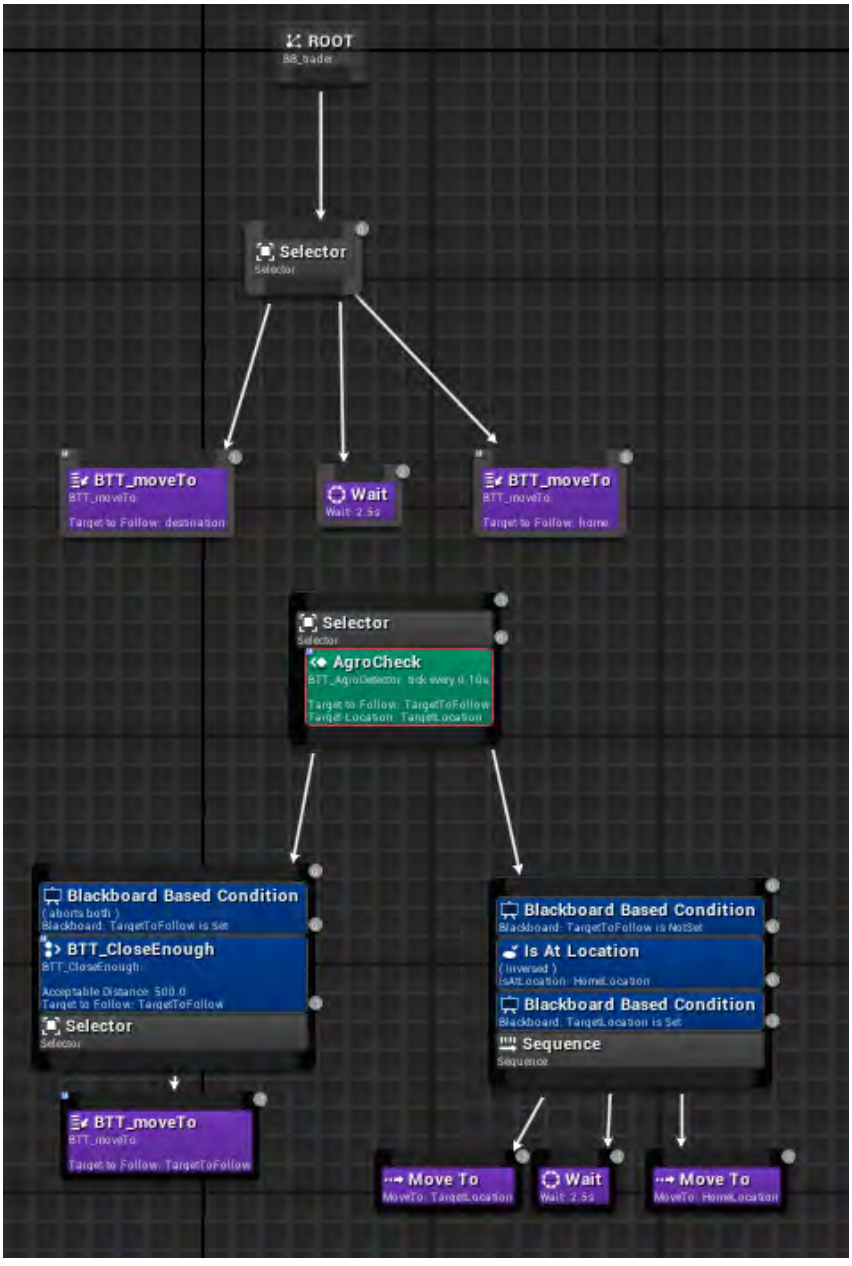


Figure 7.16: **AI Behavior Tree**. This picture shows the Behavior Tree that handle's what the AI actor attempts to do.

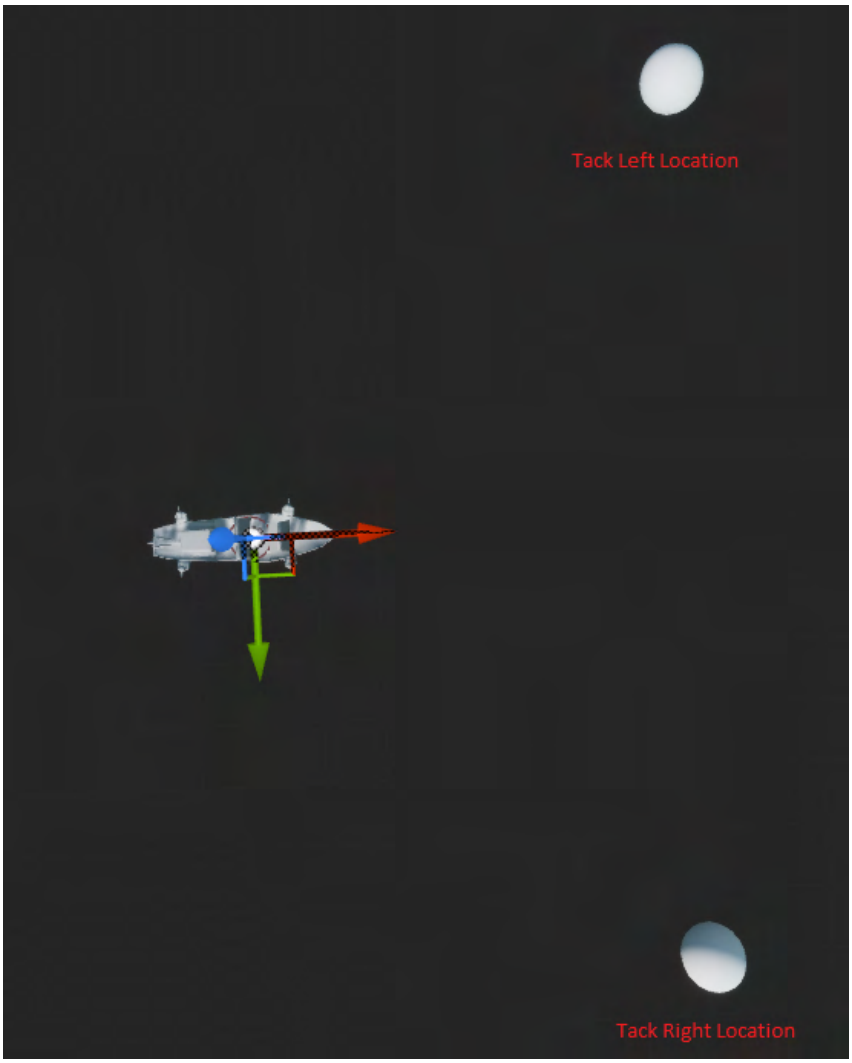


Figure 7.17: **Tacking Visualization**. This picture shows the how tacking the ship to the left or right from its main forward course.

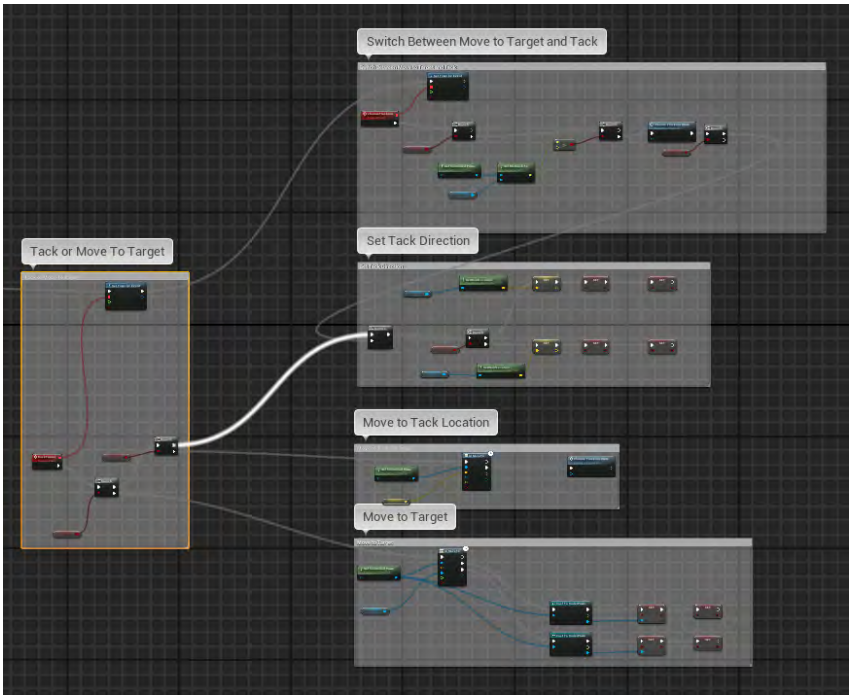


Figure 7.18: **Tacking Blueprint**. This picture shows the the actor switches between direct movement and tacking.

Chapter 8

Conclusion

In this thesis we have described a game design framework that defines a unique methodology for analyzing and evaluating player engagement in a video game. This framework is based on organizing *Player Decision Cycles* into groups of decisions that happen on similar time scales. The framework partitions the player's decision making process into three groups: (1) short term continuous decision making, which is predominant in a pure action game or FPS; (2) medium term or deliberate decision making, which is predominant in a narrative adventure game or RPG; and, (3) long term decision making, which is epitomized in the open world game where the player has complete freedom of choice in making his next quest or action decision.

The flexibility of the framework permits analysis of games at any stage in the development process. For example, the framework can be used to evaluate potential player engagement during the design phase of game development, before any code has been written; it can be used during development to gauge the engagement potential of the game as it evolves; and finally, the framework can be used after the game is deployed to facilitate a *post mortem*[1] evaluation of the game.

To illustrate the game design framework, we have designed and implemented two video games and evaluated their potential player engagement using the framework. In the case of Dragon Mist we used the framework to analyze the game design of Skyrim to locate areas of game play where complicated genetic information could be incorporated into the game without damaging player engagement. In the case of Radium Seas we used the framework to identify the basic elements of a large video game, such as Skyrim, that might be exploited

[1]The definition of *post mortem* can be found in the Urban Dictionary:

> After a project finishes in a big production for a movie, video game, or large project. Everyone gets together to discuss in a meeting what went wrong and what could be done better. The issues brought up are almost always completely ignored by management and they continue to make the same incompetent mistakes they always have.

in the design and implementation of a similar game with a smaller scope so that Radium Seas might be created by a single developer.

Our future work entails the addition of more quest lines into Dragon Mist to incorporate additional educational genetic information into the mod while continuing to enhance the information already contained in the mod, including additional dragon types, books detailing past actions taken to create new dragons, and journeys into the broader world of Skyrim to collect additional genetic samples. We also intend to expand Radium Seas to include additional menus, quest lines, graphical enhancements, NPCs, and additional facility for the player to customize the game and achieve higher levels of game play.

8.1 Acknowledgement

The development of Dragon Mist was supported by Clemson University, College of Education ADR grant, awarded to Meihua Qian (PI), Brian Malloy (Co-PI), Sam Sparace (Co-PI), and Rebecca Clark.

Bibliography

[1] Katrin Becker. Why educational games are still boring.... *The Becker Blog*, June 2010.

[2] ClassicReload.com. Number Munchers, 2017. https://classicreload.com/number-munchers.html.

[3] Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. Toward an understanding of flow in video games. *Comput. Entertain.*, 6(2), July 2008.

[4] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper Perennial, New York, NY, March 1991.

[5] Claudio Dondi and Michela Moretti. A methodological proposal for learning games selection and quality assessment. *British Journal of Education Technology*, 28(3), April 2007.

[6] Seth Fletcher. Video games: Don't be boring. *Scientific American*, February 2014.

[7] Epic Games. Unreal Engine Community Wiki, 2017. https://wiki.unrealengine.com/Main_Page.

[8] James P. Gee. *Why Video Games are Good for Your Soul*. Common Ground, April 2005.

[9] James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, October 2003.

[10] James Paul Gee. Learning by design: good video games as learning machines. *E-Learning*, 2(1), 2005.

[11] Eric Gregory. Understanding video gaming's engagement: Flow and its application to interactive media. *Media Psychology Review*, 1(1), 2008. http://mprcenter.org/review/gregory-video-game-engagement/.

[12] Mark Griffiths. The educational benefits of videogames. *Education and Health*, 20(3), 2002.

[13] Trevir Nath. Gaming will hit $91.5 billion this year - Newzoo, 2016. http://www.nasdaq.com/article/investing-in-video-games-this-industry-pulls-in-more-revenue-than-movies-music-cm634585.

[14] BBC News. Casual games make a serious impact, 2017. http://news.bbc.co.uk/2/hi/technology/7301374.stm.

[15] Justin Peters. World of borecraft. *Gaming: the Art of Play*, June 2007.

[16] Marc Prensky. Digital game-based learning. *Comput. Entertain.*, 1(1):21–21, October 2003.

[17] Brendan Sinclair. Investing in Video Games: This Industry Pulls In More Revenue Than Movies, Music, 2017. http://www.gamesindustry.biz/articles/2015-04-22-gaming-will-hit-usd91-5-billion-this-year-newzoo.

[18] Jimmy Wales. Fallout Wiki: Fallout 4, 2017. http://fallout.wikia.com/wiki/Fallout_4.

[19] Wiki. Creation Engine, 2017. https://en.wikipedia.org/wiki/Creation_Engine.

[20] Wikipedia. Fallout (series), 2017. https://en.wikipedia.org/wiki/Fallout_(series).

[21] Wikipedia. Munchers, 2017. https://en.wikipedia.org/wiki/Munchers.

[22] Wikipedia. Pirates of the Burning Seas, 2017. https://en.wikipedia.org/wiki/Pirates_of_the_Burning_Sea.

[23] Wikipedia. The Elder Scrolls, 2017. https://en.wikipedia.org/wiki/The_Elder_Scrolls.

[24] Wikipedia. The Witcher (Video Game), 2017. https://en.wikipedia.org/wiki/The_Witcher_(video_game).

[25] Jose P. Zagal and Michael Mateas. Time in video games: A survey and analysis. *Simulation & Gaming*, August 2010.