|  |
| --- |
| Implementing Drama Management for Improved Player Agency in Interactive Storytelling  Christopher McEvoy  BSc (Hons) Computer Game Applications Development, 2019 |

|  |
| --- |
| School of Design and Informatics  Abertay University |

Table of Contents

[Table of Figures iii](#_Toc6931226)

[Acknowledgements v](#_Toc6931227)

[Abstract vi](#_Toc6931228)

[Glossary vii](#_Toc6931229)

[1 Introduction 1](#_Toc6931230)

[1.1 A Brief History of Game Narrative 1](#_Toc6931231)

[1.2 Agency and Narrative 2](#_Toc6931232)

[1.3 Drama Management 2](#_Toc6931233)

[1.4 An Ideal System 3](#_Toc6931234)

[1.5 The Project 4](#_Toc6931235)

[1.6 Research Question 4](#_Toc6931236)

[1.7 Aim 5](#_Toc6931237)

[1.8 Objectives 5](#_Toc6931238)

[2 Literature Review 6](#_Toc6931239)

[2.1 Narrative Design Solutions 6](#_Toc6931240)

[2.2 Storytelling Systems 8](#_Toc6931241)

[2.2.1 Emergent Narrative Systems 8](#_Toc6931242)

[2.2.2 Drama Management Systems 9](#_Toc6931243)

[3 Methodology 16](#_Toc6931244)

[3.1 Environment and Player Interaction 16](#_Toc6931245)

[3.2 Narrative Design 19](#_Toc6931246)

[3.3 System Architecture 20](#_Toc6931247)

[3.3.1 Requirements Overview 21](#_Toc6931248)

[3.3.2 Plot Point Space Structure 22](#_Toc6931249)

[3.3.3 Player Modelling 30](#_Toc6931250)

[3.3.4 Player Action 38](#_Toc6931251)

[3.3.5 Information Replacement 39](#_Toc6931252)

[3.4 System Summary 45](#_Toc6931253)

[3.5 Other Considerations 46](#_Toc6931254)

[3.6 Data Collection and Analysis 47](#_Toc6931255)

[4 Results 48](#_Toc6931256)

[4.1 Requirements and Dataset 48](#_Toc6931257)

[4.2 Path Traces 53](#_Toc6931258)

[4.3 Performance 60](#_Toc6931259)

[5 Discussion 61](#_Toc6931260)

[5.1 Quantitative Analysis 61](#_Toc6931261)

[5.2 Qualitative Analysis 63](#_Toc6931262)

[5.3 Implications 65](#_Toc6931263)

[5.4 System Improvements 66](#_Toc6931264)

[6 Conclusion 69](#_Toc6931265)

[6.1 Future Work 70](#_Toc6931266)

[References 72](#_Toc6931267)

[Appendix A: Dataset 74](#_Toc6931268)

[Appendix B: Full History Tables 74](#_Toc6931269)

[Appendix C: Table of Processing Times 74](#_Toc6931270)

# Table of Figures

[Figure 1 – The drama management problem is to compute trajectories through state space. 11](#_Toc6930774)

[Figure 2 – Screenshot of monitor mechanic 17](#_Toc6930775)

[Figure 3 – Power puzzle mechanic example. Top: Unsolved. Bottom: Solved. 18](#_Toc6930776)

[Figure 4 – Structure of the prototype DM system. 21](#_Toc6930777)

[Figure 5 – Visualisation of the representation of plot point precondition logic. 23](#_Toc6930778)

[Figure 6 – Example of precondition logic that can only be represented by a duplicate entry of a plot point (X) in the dynamic matrix. 23](#_Toc6930779)

[Figure 7 – A visualisation of how the sets can change as points are fired. 25](#_Toc6930780)

[Figure 8 – Example of a possible graph structure before assigning endpoint dependencies. 27](#_Toc6930781)

[Figure 9 – Endpoint dependencies calculated during initialisation. 27](#_Toc6930782)

[Figure 10 – Endpoint dependency for e1 gets propagated back to B after A and C are resolved. 28](#_Toc6930783)

[Figure 11 – Endpoint dependency now assigned to C. 28](#_Toc6930784)

[Figure 12 – With different precondition logic, an endpoint dependency exists at C before other points have even fired. 29](#_Toc6930785)

[Figure 13 – With different precondition logic, multiple preconditions of X are endpoint dependencies of e1. 29](#_Toc6930786)

[Figure 14 – PM weight shifting over several steps. 32](#_Toc6930787)

[Figure 15 – Symmetrical weight changes about the mid value. 33](#_Toc6930788)

[Figure 16 – Swapping the positions of the old PM weight and the adjusting weight, the decrease in the right example is less than the increase of the left example. 34](#_Toc6930789)

[Figure 17 – PM weights can still be reinforced with adjusting weights equal to or less than the old PM weights. 35](#_Toc6930790)

[Figure 18 – No weight change when PM weight and adjusting weight are equal to the mid value. 35](#_Toc6930791)

[Figure 19 – PM weight change is affected by the distance between the adjusting weight and the mid value. 36](#_Toc6930792)

[Figure 20 – Future branch point weights become more similar to the new PM weight. This can be visualised as clustering around a point. 38](#_Toc6930793)

[Figure 21 – Diagram of how modules connect to each other 40](#_Toc6930794)

[Figure 22 – Example of a round of replacement 44](#_Toc6930795)

[Figure 23 – Example of another round of replacing 45](#_Toc6930796)

[Figure 24 – Plot Point 65 is fired, displaying its narrative text on the monitor, followed by information for the tester. 49](#_Toc6930797)

[Figure 25 – Weight change history and similarity of fired weights to player model. 50](#_Toc6930798)

[Figure 26 – Set change history as points in the possibility space are made active or closed. 51](#_Toc6930799)

[Figure 27 – Change history within the active set as points move from the W-Set to the !W-Set and vice versa. 51](#_Toc6930800)

[Figure 28 – The current undiscovered, closed, W and !W sets. 52](#_Toc6930801)

[Figure 29 – Currently non-closed endpoints and their current dependencies. 52](#_Toc6930802)

[Figure 30 – Path A: Shortest Path to Endpoint 69 53](#_Toc6930803)

[Figure 31 – Path B: Shortest Path to Endpoint 71 54](#_Toc6930804)

[Figure 32 – Difference between Paths A and B 55](#_Toc6930805)

[Figure 33 – Path C: A Longer Path to Endpoint 71 55](#_Toc6930806)

[Figure 34 – Path D: Endpoint 86 56](#_Toc6930807)

[Figure 35 – Path E: Endpoint 96 56](#_Toc6930808)

[Figure 36 – Path F: Endpoint 102 57](#_Toc6930809)

[Figure 37 – Path G: Endpoint 148 58](#_Toc6930810)

[Figure 38 – Path H: A Longer Path to Endpoint 148 58](#_Toc6930811)

[Figure 39 – Path I: No Path Found 59](#_Toc6930812)

[Figure 40 – Average PM Difference as Path Length Increases 59](#_Toc6930813)

# Acknowledgements

I would like to take the opportunity to thank several people for their support throughout this project, and throughout the duration of my time at Abertay University.

Firstly, I would like to thank my supervisor, Dr David King, for his assistance and guidance during the project.

I would like to thank my family for their support during my studies.

Additionally, I would like to thank my friends who have encouraged and supported me throughout my final year project, especially Robin Lawrence who worked next to me, keeping me motivated and more importantly, sane.

# Abstract

**Context/Background:** With increases in complexity of graphics in video games, there exists a need to increase the complexity of game world narratives so that players feel they are an active part of an unfolding story, influenced by their actions and behaviours. Drama Management systems offer an attempt to facilitate this but are an area in need of further exploration for application in real-time narrative games.

**Aim:** The aim of the project is to develop a prototype Drama Management system for a real-time game that improves player agency, while also reducing the problem of trade-off between designer intent and player agency, and to analyse the effectiveness of the chosen techniques.

**Method:** An application was developed consisting of a 3D interactive environment, a possibility space of narrative plot points, and an Intelligent Agent that decides how to branch the story based on a predicted Player Model, using Heuristic Search Planning.

**Results:** It was determined that the possibility space design has a major role in the application’s effectiveness to invoke agency within players. The framework created is transferable to other projects for further modification specific to realising their individual goals. The sense of agency can also be improved by combining the developed framework with additional extensions.

**Conclusion:** This project determined that Drama Management systems are a viable method of improving the complexity of a narrative’s discourse to promote player agency, but also require very careful design alongside suitable algorithmic techniques in order to be fully effective.

**Keywords:** Drama Management, Artificial Intelligence, Intelligent Agents, Player Modelling, Heuristic Search Planning, Graph Theory

# Glossary

AC – Affective Computing

CBR – Case-Based Reasoning

D&D – Dungeons and Dragons

DDM – Distributed Drama Management or Manager

DM – Drama Management or Manager

HCI – Human-Computer Interaction

HSP – Heuristic Search Planning

IA – Intelligent Agent

NPC – Non-Playable Character

OCC - Ortony, Clore and Collins

PM – Player Model or Modelling

POCL – Partial Order Causal Link

POP – Partial Order Planning

RBS – Rule-Based System

RL – Reinforcement Learning

RPG – Role-Playing Game

TTD-MDP – Targeted Trajectory Distribution Markov Decision Processes

UX – User Experience

# 1 Introduction

## 1.1 A Brief History of Game Narrative

Early video games were able to show real-world depictions with very limited resources. Lacking a Von Neuman architecture, games such as *Tennis for Two* (Higinbotham, 1958), *Spacewar!* (Russell et al. 1962) and *Pong* (Atari, 1972) did not have working memory, and so were limited in their visual representation of in-game elements.

Since the invention of the microprocessor in 1971, game behaviour and audiovisuals have continuously grown in complexity. Games could become more detailed and expressive in appearance. For example, the vibrant colour-changing experience of *Centipede* (Atari, 1980) only became technically-feasible with this advancement. With more available resources also came the possibility to include narrative elements.

*Adventure* (Atari, 1980) is arguably the first video game of the adventure genre, taking its influence from the text-based interactive fiction stories that came before it, and the table-top role-playing-game (RPG) phenomenon *Dungeons and Dragons* (D&D) (Gygax, 1974) that appeared midway through the 70s. *Donkey Kong* (Nintendo, 1981) is arguably the first game with a storyline that utilises visuals to convey a narrative, as it tells a story of Jumpman trying to save Pauline from a large gorilla.

From these formative years, a link has been established between the complexity of computer graphics and the complexity of game behaviours. With modern games, there exists a need for these elements to be comparable in complexity. A game with a very detailed game world but basic behaviours for game agents can lead to players perceiving those agents as ‘dumb’, shallow or out-of-place; reducing the player’s enjoyment of the experience. Similarly, the complexity of game narratives has also been increasing to match these other elements.

## 1.2 Agency and Narrative

Game AI complexity has been continually increasing to match the complexity of modern game graphics. The goal has been to provide players with an immersive experience where Non-Playable Characters (NPCs) behave in a believable way. Applications of this have included the creation of AI opponents that can adapt to the player’s behaviour, creating more challenge. In the context of narrative games, this adaptability is also desirable but for constructing stories in which the player has a greater perception of agency.

Agency is the degree of influence a player-character’s actions have on the state of the game world. Thus, narrative agency is much desired by players as they can meaningfully change the story and feel that their choices matter (Hernandez et al. 2014). Agency is important to an interactive narrative as it adds to the believability of the situation. Thus, there is a correlation between player agency and immersion, an element that contributes to the enjoyment of games.

Narrative provides a fundamental aspect to modern video games, providing context to the detailed visuals and actions the player is asked to perform, with agency playing an important role in the enjoyment of the narrative experience.

## 1.3 Drama Management

There is a growing research community for the application of Artificial Intelligence systems for interactive storytelling. This area is expansive, with many applications and subsets, such as narrative generation, natural language understanding, and drama management (Mateas et al. 1999). Storytelling is a complex task that this field of research seeks to model.

Drama Management systems fall within this area of research. Rather than generating stories it manages the selection of story content based on the player’s actions (Reidl et al. 2012) using a Drama Manager (DM) to guide the story along a changing narrative arc. They are most often used to select and order story content created by a human author. DM systems vary in methodology while sharing many design challenges including: the trade-off between designer intent and player agency (Reidl et al. 2012), and the ‘boundary problem’ where a player’s actions fall outside the scope of authored content (Sharma, et al. 2010). The boundary problem occurs where a player does an action that the system is not prepared for, which can lead to an illogical or erroneous response by the system.

## 1.4 An Ideal System

To address some of the issues with narratives in complex authored story spaces, an ideal system would incorporate positive attributes of conventional branching narrative design and DM system design, so that the story experience is tailored to the individual player while increasing the effectiveness of the illusion of choice, making it more difficult to notice the deception; and therefore improve player agency with reduced drawbacks. A more flexible system would apply constraints without reducing the player’s perceived range of freedom of action, limiting not what the player can do but what they are likely to think of doing next (Sharma et al. 2010).

In process-oriented games, players are given a system to play with rather than a goal given to the player by the designer (Nielsen et al. 2008). Such games allow players to create their own emergent goals to aim for. *The Stanley Parable* (Wreden et al. 2013), described as a ‘Narrative Sandbox game’, uses a lack of designer-set goals in a true branching narrative to allow player choices to be expressive and meaningful without contributing to a larger objective. The game’s creators argue this makes them reflective of the players themselves (Wreden et al. 2014). This also provides space for players to set themselves goals in a similar way to process-oriented games. This concept of player-set goals can be applied using a DM in narrative games without true branching structures to improve player agency in more complex story spaces. However, the concept of using a DM that facilitates greater freedom of action for players to pursue their own goals while still guiding towards an authored dramatic arc, has been unexplored in commercial games.

## 1.5 The Project

The complex phenomena of storytelling make it impossible to create a decisively-testable model in the same way as conventional AI engineering (Mateas et al. 1999). This makes it an area worth researching. There is no complete solution to the issue of narrative in games, but a lot can be learned from the approach of film theory. By selectively ignoring some standard conventions, new conclusions can be drawn and taken forward in future projects. The purpose of the project is to investigate the use of Intelligent Agents (IAs), in the form of a DM system, and techniques such as Heuristic Search Planning (HSP) to improve player agency in complex story spaces.

Previous approaches to DM systems have largely constrained the scope of the solution to that of a somewhat limited possibility space. Additionally, very few of these solutions have been implemented in a 3D environment suitable for a real-time interactive storytelling experience. Instead, these tend to be achieved in the form of text-based games. However, this is arguably insufficient for demonstrating the viability of DM systems in modern video games, which are mostly audio-visual.

This project seeks to provide a sizable possibility space within a real-time 3D environment to investigate the viability of DM systems in video games with larger and more complex environments and methods of player interaction. The solution acts as a test for scaling to larger projects.

## 1.6 Research Question

How can Drama Management and Player Modelling techniques be used to improve player agency in complex narrative environments?

## 1.7 Aim

The aim of the project is to develop a prototype Drama Management system for a real-time game that improves player agency, while also reducing the problem of trade-off between designer intent and player agency, and to analyse the effectiveness of the chosen techniques.

## 1.8 Objectives

* To assess current narrative intelligence techniques. In particular, to assess techniques focusing on the selection of designer-authored story content, in Drama Management systems,
* To build an application that combines multiple appropriate techniques, with consideration of conventional interactive narrative design,
* To evaluate the effectiveness of the created application, and make recommendations, and suggest future work.

The format of the dissertation will be as follows:

Chapter 2 discusses previous work in Drama Management, including various combinations of techniques used in previous approaches.

Chapter 3 discusses the methodology undertaken for this project that stems from these previous approaches.

Chapter 4 discusses the qualitative and quantitative results of testing, including the approaches to obtaining data.

Chapter 5 discusses the effectiveness of the solution, including the significance of the project to this field of research.

Chapter 6 is a conclusion that outlines suggested further work and development.

# 2 Literature Review

This section discusses previous approaches to technical solutions for improving the narrative experience in interactive fiction. These technical solutions tend to fall within two main classes; Emergent Narrative Systems and Drama Management Systems, but this distinction is blurred as techniques are combined. The project focuses on DM systems as this class is arguably more transferable, at the present time, to modern commercial games that have designer-authored story content. This section begins with a description of design-based approaches to game narrative, as this will inform the requirements for the project and the technical solutions available.

A narrative can be defined as a structured series of events (Roberts et al. 2009). Interactive narratives involve player participation in the story world, allowing them to change it in meaningful ways. With interactivity comes new problems with storytelling that require both design solutions and technical solutions. There have been several different approaches to improving the complexity of interactive narratives to improve agency. These approaches fall into several categories, many of which have been combined with DM systems for different purposes.

## 2.1 Narrative Design Solutions

Game narratives generally fall into two main classes, linear games and branching games. Linear games are those with a set structure that do not contain choices made by the player that impact the world in a meaningful way. These games have the least potential agency, although this does not mean they do not deliver a good narrative. Linear game narratives can still facilitate immersion by increasing the level of detail of the environments, characters and dialogue. This approach works well, as seen in games such as *The Last of Us* (Naughty Dog, 2013) but is usually restricted to larger AAA studios with the resources to create such sprawling worlds. Smaller development teams without this ability may need to rely on other methods to improve player agency.

The desire for player agency has led many games to take a branching approach to the narrative structure of their stories. In such story worlds, the player’s actions or decisions can lead to a different conclusion, or a different series of plot points. However, there exists numerous issues still unsolved concerning branching narratives. First, the ‘illusion of choice’ is where a player’s choices may branch the story but ultimately have little or no impact on the conclusion. At the time, the player’s choice may appear to have meaning but once a player recognizes that their choice has less meaning than initially perceived, they experience a reduction in agency.

True branching narratives partially avoid this issue by having many choices contribute to the final outcome, but such structures have additional problems. They can grow exponentially, making the task of asset creation and story authoring more challenging, increasing development time. Additionally, players may still perceive their actions as meaningless in true branching structures as employing a rigid choice system can make the player feel that no matter what choice is made it is ultimately enacting the designer’s will, with no real freedom for the player, again resulting in a decrease in agency. Finally, branching narratives can make replayability feel tedious for players trying to experience every possible narrative.

There are also games that exist somewhat like a hybrid between linear and branching games, where the outcome or ending of the story is determined by some hidden ‘score’ value rather than specific individual choices. In these ‘outcome-oriented’ games, the narrative structure is typically linear, with the score value being calculated by small choices in dialogue options or decisions that only contribute to the story conclusion. The value may also be determined by a player’s playstyle as seen in *Dishonored* (Arkane Studios, 2012), where the outcome of the narrative is determined by how the player dispatches enemies, using lethal or non-lethal methods. With this structure however, because the effects are usually only shown at the very end of the story, it can make the rest of the choices in the plot up until that point seem comparatively meaningless.

It is worth noting that these structures are not restrictive, and any combination of these structures can be used, to varying degrees, and in a seemingly endless number of ways. For interactive fiction games using artificial intelligence techniques, such as Intelligent Agents, narrative design methodology and structure still plays a crucial role.

## 2.2 Storytelling Systems

Storytelling systems fall within two main classes: Emergent Narrative systems, and Drama Management systems. Where Emergent Narrative systems are simulations constructed from Intelligent Agents, Drama Management systems use a single Intelligent Agent, the Drama Manager (DM), to monitor the game world and drive the authored story forward based on the player’s actions (Reidl et al. 2012).

Individual approaches may borrow or combine traits from either class. A specific implementation may be considered to fall on a spectrum between prioritising player spontaneity and enforcing an authored narrative arc (Martens et al. 2017). As mentioned earlier, this manifests as a trade-off between designer intent and player agency, which can become a problem. Compelling narratives can effectively balance these attributes.

### 2.2.1 Emergent Narrative Systems

Emergent Narrative systems are rich simulations underlying the game world that seek to generate unique stories through Intelligent Agents.

Intelligent Agents in games are NPCs that have some degree of freedom to act autonomously. They have goals and the ability to react to sensory information. Narratives then emerge from the interactions between IAs and the player, the world, and each other. The designer of emergent narrative systems creates the rules for interaction and rules for changing the agents’ properties. The player can then interact with this system through actions and receive responses. IA characters can also have personalities and relationships, as well as other characteristics that influence their behaviours.

Intelligent Agents can increase the complexity of the simulated world of a video game to give a richer experience. For example, strategy games can use adaptable opponents to challenge the player. With narrative games, IAs can add complexity to the story. Systems using IAs for interactive fiction present choices to the player that are generated by the simulation (Martens et al. 2017). These systems use a story generation algorithm, with designer inputs to initialise the simulation, and allow players to take improvised actions and have the system react in meaningful ways.

One benefit of such systems is that the player can feel a sense of ownership, that they are the only person who will experience an individual moment of their story experience. However, when written poorly, these systems tend to be severely limited in their storytelling capabilities; feeling repetitive, basic and unstructured, even if they can still sometimes generate an interesting situation. At this time, no story generation algorithm can deliver interactive narrative as well as a skilled writer, and emergent narrative remains an ongoing research area. Because of these limitations of emergent systems, the focus of this project is on the other class of system, Drama Management systems, as these involve wholly designer authored content and focus on delivering structure.

### 2.2.2 Drama Management Systems

Emergent Narrative systems have previously incorporated elements of DM systems in their design, creating a hybrid between the two system types. However, conventional DM systems are typically concerned with the structured ordering of authored story content rather than that of generated content. In such systems, a Drama Manager is used to perform the story planning. A Drama Manager is typically a single IA that makes decisions based on logical criteria and one or more additional metrics.

#### *****Current Techniques*****

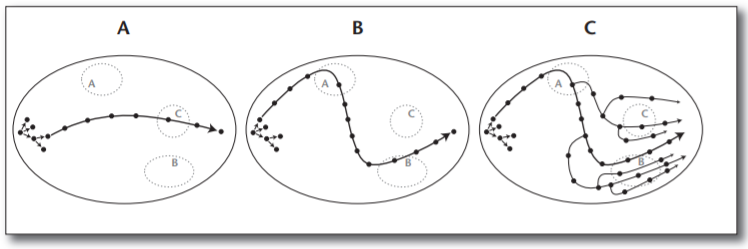
While DM systems vary in methodology, they all operate on similar principles to achieve different, but not dissimilar, goals. Events in a linear narrative can be called plot points, which are arranged sequentially. In non-linear games there can exist a state-space of plot points. DM systems use various techniques for constructing or ordering narrative arcs between plot points.

Many systems define plot points with two attributes: preconditions and effects. Preconditions describe what is required for the plot point to be accessible, and effects describe what changes are made to the story world by the plot point. These preconditions and effects connect plot points into an acyclic directed graph of nodes, and this format is used when planning the plot discourse.

DM systems employ different techniques depending on their goals. A common goal is the structured ordering of a narrative arc that follows a plot progression model and accounts for interactivity, using a planning algorithm. Some systems have made use of some form of Trajectory Space Search such as Partial Order Planning (POP), Targeted Trajectory Distribution Markov Decision Processes (TTD-MDP), and Partial Order Causal Link (POCL) plans.

With these techniques, the state space of plot points becomes a trajectory space. Trajectory space is like state space, except every point in the space is a partial or complete trace of the game (Roberts et al. 2009). The nature of trajectory space makes it more computationally expensive, with a larger search space, but allows the DM to plan. However, the DM cannot plan by itself, it needs at least one other module to inform the decision-making, where a module is a separate component that contributes to the overall system. Using information from its other modules, the DM can plan so that future problems can be avoided or to further enhance the story experience by selecting the most preferable path.

To select a plot point from a series of possibilities, many DM systems use a form of Heuristic Search Planning to evaluate the best plot point, or trajectory, for a given situation. The heuristic can be calculated based on several attributes usually returned by another module to the DM.



**Figure 1 – The drama management problem is to compute trajectories through state space.**

a. A possible narrative trajectory through state space. b. A possible narrative trajectory that visits plot points deemed favourable and avoids plot points deemed unfavourable. c. Accounting for player interaction.

Figure 1 above (Roberts et al. 2011) is a visualization of trajectory space, as a directed graph, where each dot represents a plot point. There can exist a very large number of possible trajectories that, based on player interaction, are guided towards desirable plot points. Example A shows an example of a path through the possibility space. Example B shows how a system can direct the path towards more favourable plot points and away from unfavourable points. Finally, example C shows how an interactive system with user input can lead to branches that navigate the space differently. HSP can be used to decide how favourable points are, and the plot can be branched using the heuristic and actions taken by the player.

The heuristic used to select plot points can be based on output from another module that communicates with the DM. Previous approaches have calculated the heuristic using a plot progression model, while other approaches use various forms of player modelling (PM). Player modelling has been used to model several different metrics through which a DM system can decide the possible future trajectories. Player models can be constructed through direct measurements using physical devices to measure heart rate and galvanic skin response or, less intrusively, indirect measurements such as order of actions and actions-per-minute to infer information about the player. DM systems incorporating player modelling attempt to model the player’s state explicitly and shape the narrative specifically to influence it (Hernandez et al. 2014). The fields of Affective Computing (AC) and Cognitive Science are closely related to the challenges of developing artificial intelligence systems for interactive narrative, making them good areas to examine when developing a DM solution that incorporates player modelling.

Player models have been based on a player’s predicted emotional state. Others have been based on their predicted playstyle. Playstyle model approaches have been used to determine how experienced a player is and if they need hints, while other approaches use the playstyle to infer a predicted player goal, using said predicted goal to predict the player’s next set of actions, which is then used to change the state of the world in preparation for these actions.

For a playstyle prediction model, one method is to annotate player actions with a set of weights to several playstyle classes created by the designer (Weyhrauch et al. 1997). This annotation is to inform the system of what playstyle the player is likely to be using. This predicted playstyle is then used to infer the predicted goals. The ordering of actions can then change the weights of the annotations, making a series of actions contribute more to one or more playstyles.

In essence, the way a player interacts with the world is used to determine what they are trying to accomplish, allowing the DM to consult the plot progression model and/or other metrics and use the result to decide what to do. The difficulty of this method is the balancing of the weights and player model algorithm to improve the accuracy of prediction. Additionally, the earlier predictions of player goals are more likely to be incorrect but improve in accuracy with a greater number of actions taken. This approach shares similar principles as reinforcement learning (RL), without the training results being retained.

#### *****Previous Work*****

Previous research and applications in Drama Management have involved the use of Adversarial Search and POCL to structure a dramatic arc while avoiding problems (Roberts et al. 2009), Case Based Reasoning (CBR) to learn how interested participants are in different sections of the plot (Sharma et al. 2010), and player modelling for suspense (Reidl et al. 2011) to name a few. This field of research also considers elements from different disciplines such as psychology and conventional narrative theory. Additionally, natural language processing has been used in several applications with text-based interfaces. Most notably, *Façade* (Procedural Arts, 2005) allows players to communicate with the game’s Intelligent Agents using natural language through text. DM systems such as that seen in *Façade* have also incorporated elements of narrative generation to extend the range of possible plot points and reduce the risk from the boundary problem.

Other approaches have used DM systems in conjunction with Intelligent Agents to add depth to an authored narrative, or to facilitate improvisation within a partially generated narrative space. One research example of such an approach is a Distributed Drama Management (DDM) system (Weallans, 2012) which seeks to retain the Emergent Narrative system benefits of believability and improvisation while still providing emotional and structural consistency.

A commercial example of a hybrid system can be found in the game Haunt 2, which uses player prediction modelling to inform a Drama Manager, to determine if the player’s actions will endanger the plot so that the player’s actions can be avoided or rejected (Magerko et al. 2004). Unlike similar DM systems, Haunt 2 uses additional timing constraints to inform the DM of how to order events. It makes up for its less-sizable range of possible plot orderings by having semi-intelligent actors that operate autonomously until the DM gives them prompts. Much like the DDM system, the DM in Haunt 2 doesn’t act directly on the world but attempts to change the world through the actors. This was effective in improving agency by further hiding the internal operation of the system from the player.

Techniques from story generation, recognition and evaluation have also been incorporated into DM systems or would make appropriate additions to a DM system. For certain thematic genres of interactive stories, a computational framework of suspense and dramatic arc (O’Neill et al. 2011) could be incorporated to assist in the selection of plot points, with a target ‘suspense value’ as the heuristic of the state search. Emotion recognition models could also be incorporated to search for target emotion states for more than just suspense. This form of emotion-recognition would need an emotion model that is in a computer-implementable format, such as a variation of the Ortony, Clore and Collins (OCC) model of emotions (Steunebrink et al. 2009). The OCC model is a hierarchical structure that describes “valenced reactions to emotions”, leading to the classification of 22 emotion types specific to the ‘consequences of events’, ‘actions of agents’ and ‘aspects of objects’. This specificity to events and perceptions makes the OCC model a powerful prospective tool for developing emotionally-intelligent agents but may need clarifications to some ambiguities for it to be viable to Intelligent Agent researchers.

There have been countless game narratives, both linear and branching, that generate excellent immersion without the need for complex algorithms, generation and authoring. The interactive narrative problem is as much of a design problem as it is a technical problem. The purpose of the research is to investigate possible improvements to immersion that exist beyond the scope of conventional interactive narrative design. The developed prototype will attempt to mitigate some of the issues of branching narratives while seeking to make improvements on previous DM approaches.

To summarise, DM systems are made up of a structure of plot points, where each plot point has a number of preconditions and effects. The DM decides which plot points are accessible in the game world, shaping the direction of the unfolding story in a structured order. The DM is informed by one or more other modules so that a heuristic can be used to decide which points to make available. Other modules have been based on plot structure, an emotional model, and predicted player models of goals. Many other techniques have been combined with DM systems to make improvements to the storytelling experience. In the next chapter, the precise methodology chosen for this project is discussed.

# 3 Methodology

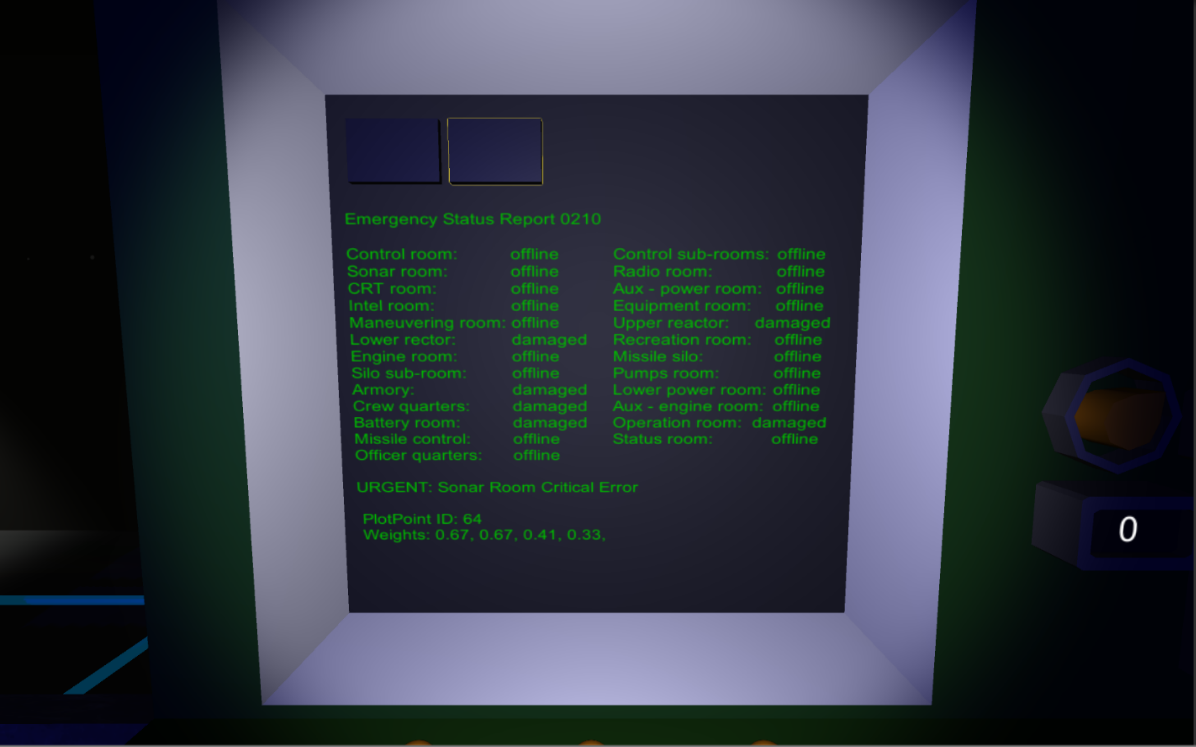
The approach taken for this project consists of several different components based on the requirements and goals. First, a 3D environment with limited methods of player interaction was developed. The second component is the narrative itself, which needs to be expansive enough to form a suitably sizable possibility space. The formal representation of this possibility space is considered as another module that the DM communicates with during its decision-making. Lastly, the third main component of the system is the system architecture, which itself consists of several modules including the Drama Manager, the Player Modeller, a connection to the Possibility Space, and an Enactor which performs world actions through the Drama Manager. The Enactor acts as an intermediary between the DM and the game world and performs the plot point ‘replacement’ as discussed later in this chapter. The structure of this is seen in Figure 4 later in this chapter.

In this chapter, these different components and modules are discussed in detail, including how they operate and how each of these three main components is subdivided. Design with respect to player psychology and system usability is also considered throughout. The chapter then summaries the entire system before discussing additional considerations and describes the approach taken to data collection and analysis.

## 3.1 Environment and Player Interaction

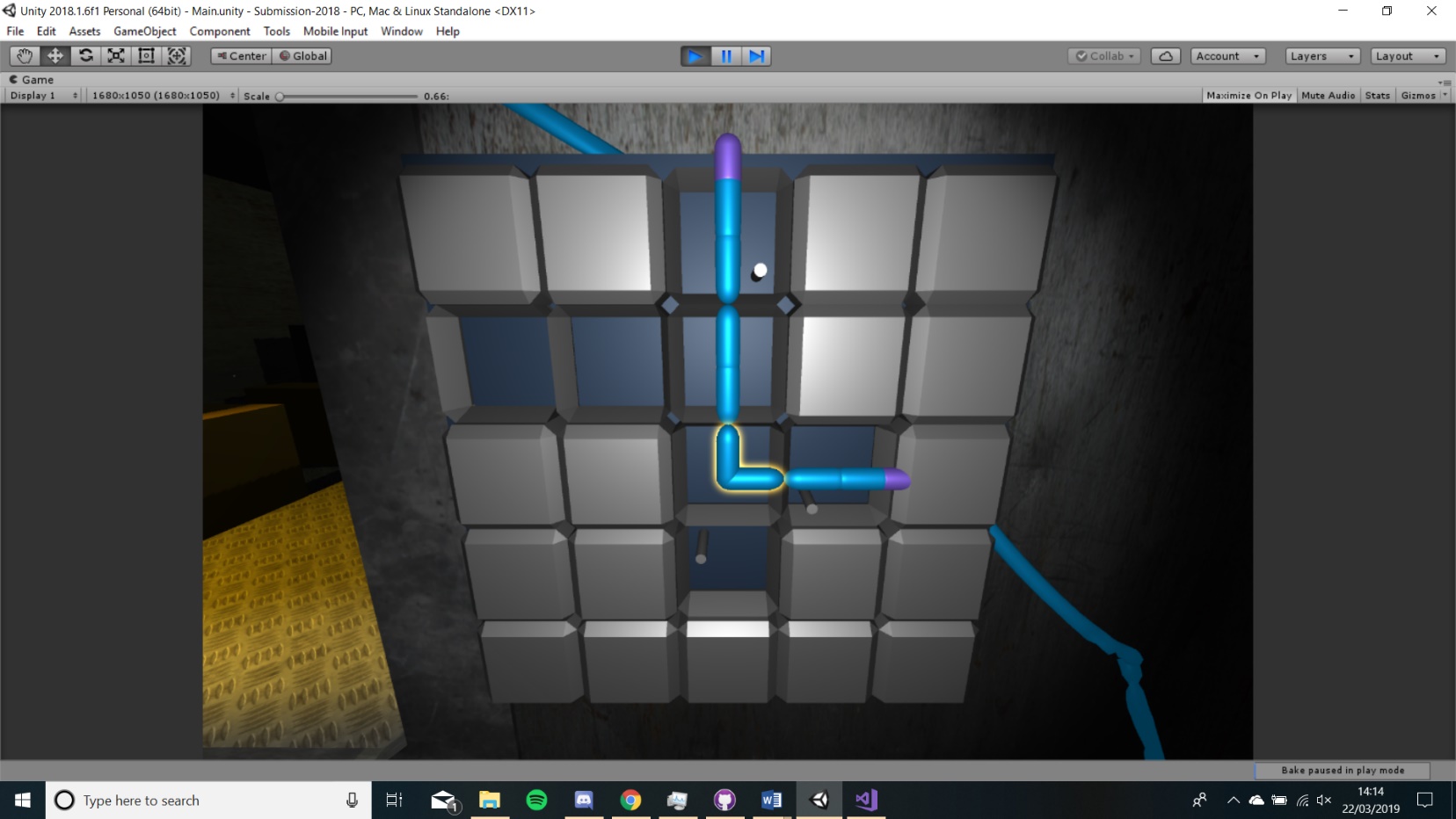
Previous approaches to artificial intelligence systems for narrative have largely existed in purely text-based format, with few exceptions. This project seeks to explore how DM systems can be used in a 3D environment where players are given a much greater freedom of choice. Players are given a much greater range of potential actions by having a sizable environment to explore, where each room has potentially-significant information. This design also allows for players to selectively explore different places, leading to a specific order of player actions, which is then used by the DM to make decisions on what direction to take the plot. The range of choice and possibility for exploration makes the interactive story appear to be unfolding more naturally, improving the illusion of choice.

The environment consists of two floors of several rooms. Each room contains one or more ‘machines’ that the player can interact with. Following verbal or written instructions, the player can use these machines to change the state of the room, represented by a colour. This is one possible player action that the DM considers. There are also several monitors in each room that the player can interact with to read information. This is an action for consideration as the information the player knows will affect future parts of the story. Figure 2 below shows a screenshot of the monitor mechanic.



**Figure 2 – Screenshot of monitor mechanic – including debug information**

There are additional design considerations that were needed to make such a narrative environment effective. The environment has restrictions imposed on the player from the start. All rooms begin closed until the player solves an electricity-themed puzzle to open doors and gain access to the machines and digital monitors inside. Figure 3 below shows an example puzzle. Without restrictions, the player could freely move from room to room without reward which gives no incentive or challenge to search for potentially-important information and can also disrupt the story structure entirely, making it difficult to become emotionally-invested. The level design, puzzle design, interaction design, and audio-visual design all impact an interactive and visual medium, so each needs some consideration as without these attributes the experience would lack depth.



**Figure 3 – Power puzzle mechanic example. Top: Unsolved. Bottom: Solved.**

## 3.2 Narrative Design

Before discussing the technical implementation, a description of the narrative design practices taken is required to understand how the DM uses the possibility space and how the DM’s decisions affect the game world. Additionally, it is useful to define the requirements of this narrative possibility space.

In most interactive narrative games, there are explicit choices presented to the player. For example, choosing between two characters to save in *The Walking Dead* (Telltale Games, 2012) video game series. This can be represented as an ‘A or B choice’ and is presented to the player in a direct manner. The player will likely make this choice based on which character they prefer as well as their perceived usefulness. This project defines such a player-decision as a ‘hard choice’; a choice that the player explicitly makes to move towards a goal created by the player or the game itself.

This narrative design goal of the project is to decide the discourse of the plot based on the player’s behaviour when navigating and interacting with the environment. A player may choose to venture off the directed path to explore and investigate in their own unique way. Their individual approach is taken in pursuit of their own goals and is based on their unique perceptions and opinions of their situation. To this effect, the player’s actions and order of these actions is what branches the story.

This project defines each step of a player’s approach to exploring a narrative space in pursuit of a goal as ‘soft choices’. In the prototype’s environment, a ‘soft choice’ example would be a player choosing to read a page from a monitor in a room. They have not been told to do so by the game, but they chose to take this action anyway. An example of a ‘hard choice’ in this environment could be where they are told explicitly to access a monitor before they decide to interact with it.

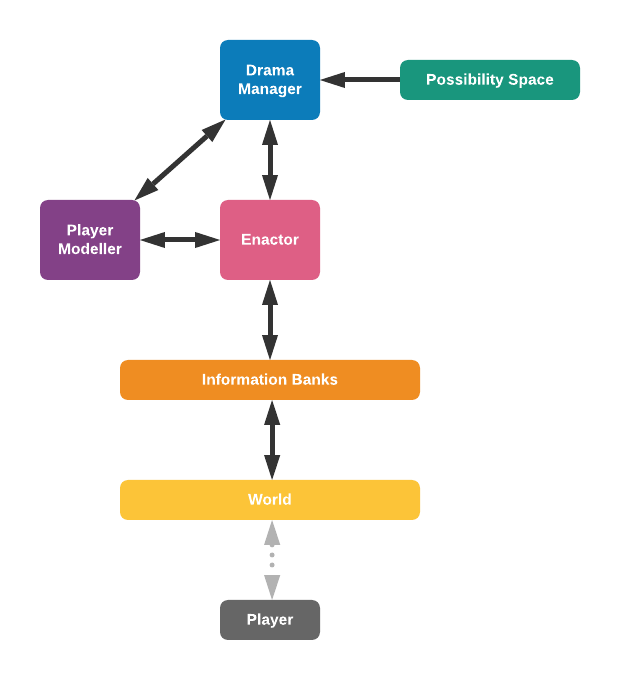
In the prototype game, when the player makes a ‘soft choice’, they are given some information. This information can uncover more of the story and give the player ideas for next actions to take. The Intelligent Agent DM will need to consider what ‘soft choices’ are made and in what order, to decide what plot points are most appropriate to present to the player.

This design approach may further improve the illusion of choice, while delivering a narrative discourse that is more specific to the individual player, leading to an improvement to player-agency.

## 3.3 System Architecture

This section begins with an overview of the system structure before giving an overview of requirements followed by discussing the implementation. The Intelligent Agent System for this project consists of several modules.

The Drama Manager takes plot points from a Possibility Space that represents an acyclic directed graph and navigates the state space. The DM handles the condition logic of the plot points and maintains a set of which points are reachable. The Player Modeller is used to keep and update a model of the player’s playstyle in the form of a set of weights. It also updates the weights of future plot points after a point is fired. The Enactor module provides the interface between the DM and the world. The Enactor performs the plot point replacement, changing what points are accessible to the player within the world. The narrative information and corresponding annotated weights are stored in several Information Banks which are accessed through player actions in the world.

****

**Figure 4 – Structure of the prototype DM system.**

Figure 4 above shows the structure of the system. The DM, PM and Enactor modules communicate with each other. The Possibility Space provides the directed graph to the DM for it to navigate. The Enactor accesses and updates the Information Banks, which in turn, relay the information to the player when accessed and pass upwards through the system where the plot point is fired by the DM.

### 3.3.1 Requirements Overview

Previous approaches to DM systems rely on ‘hard choices’ and are usually delivered through a text-only interface, eliminating or simplifying factors of the player’s location and range of actions. These approaches scale rather well but the ‘hard choices’ are delivered explicitly which can keep the player feeling constrained to bounds set by the designer.

The intention of the prototype is to provide a wider range of ‘soft choice’ options so that the player perceives a greater range of freedom, even with constraints still imposed. Because of this, the prototype must be able to handle a large range of actions to facilitate a complex narrative.

The system must be able to navigate a possibility space of plot points, using conditions to determine the range of points that are potentially available for the player to access. The system must be scalable so that many points can be added to the collection without drastically impacting performance as the game must be able to run in real-time. The DM system must decide what plot points to make available in the world based on an evaluation using a Player Model. Some actions and plot information will only make sense to be available in certain places in the environment and this will need to be factored into the plot point replacement. Finally, the system should be versatile so that it can be expanded upon in the future.

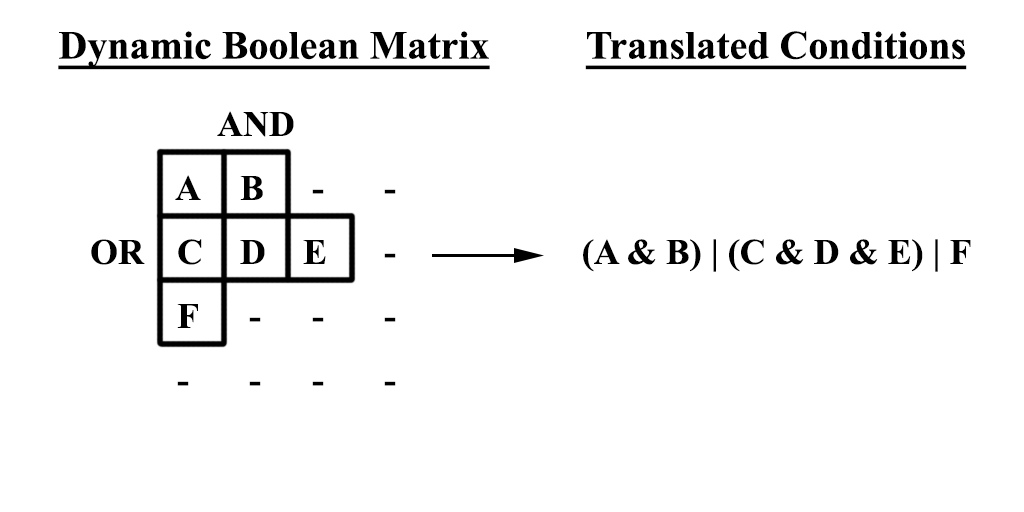
### 3.3.2 Plot Point Space Structure

The Possibility Space describes an acyclic directed graph of plot points. Its class acts like a database of these points and handles their initialisation before passing them to the DM.

Each plot point has preconditions and effects as seen in previous approaches. For a point to be up for consideration when deciding what points to put in the world, its preconditions must be met, specifying what plot points should have been accessed previously. The preconditions essentially describe a Boolean condition. Effects describe points that would become impossible if a point was accessed; points that would no longer make sense in the narrative; effects are two-way so that accessing A could make B impossible and vice versa.

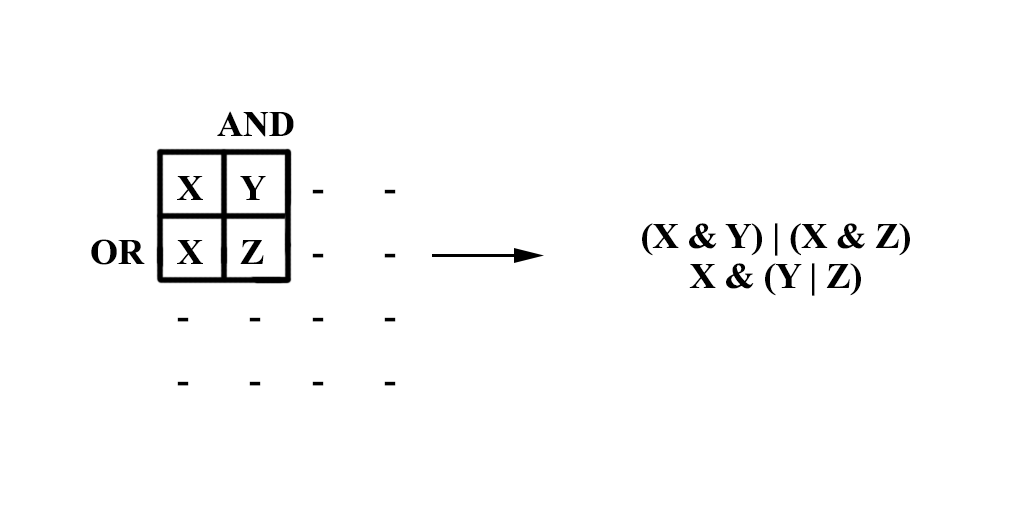
#### *****Dynamic Boolean Matrix*****

The Boolean condition of the precondition logic can be more complex than a simple ‘AND’ or ‘OR’ statement. There may be many conditions required and many logical operators. This project solves this problem by creating a method of defining condition logic, such that statements of any length and combination of ANDs and ORs can be specified. This project refers to the created solution as a ‘dynamic Boolean matrix’.

****

**Figure 5 – Visualisation of the representation of plot point precondition logic.**

Figure 5 above shows that plot points in a row are ANDed together, and the result of each row is ORed with all other rows. The figure shows how this translates to a non-trivial Boolean statement. The matrix does not incorporate NOT operators as these are represented by the effects.

****

**Figure 6 – Example of precondition logic that can only be represented by a duplicate entry of a plot point (X) in the dynamic matrix.**

There are cases where achieving a desired logical condition requires duplicate entries to be made in a point’s matrix. Figure 6 above shows an example of this.

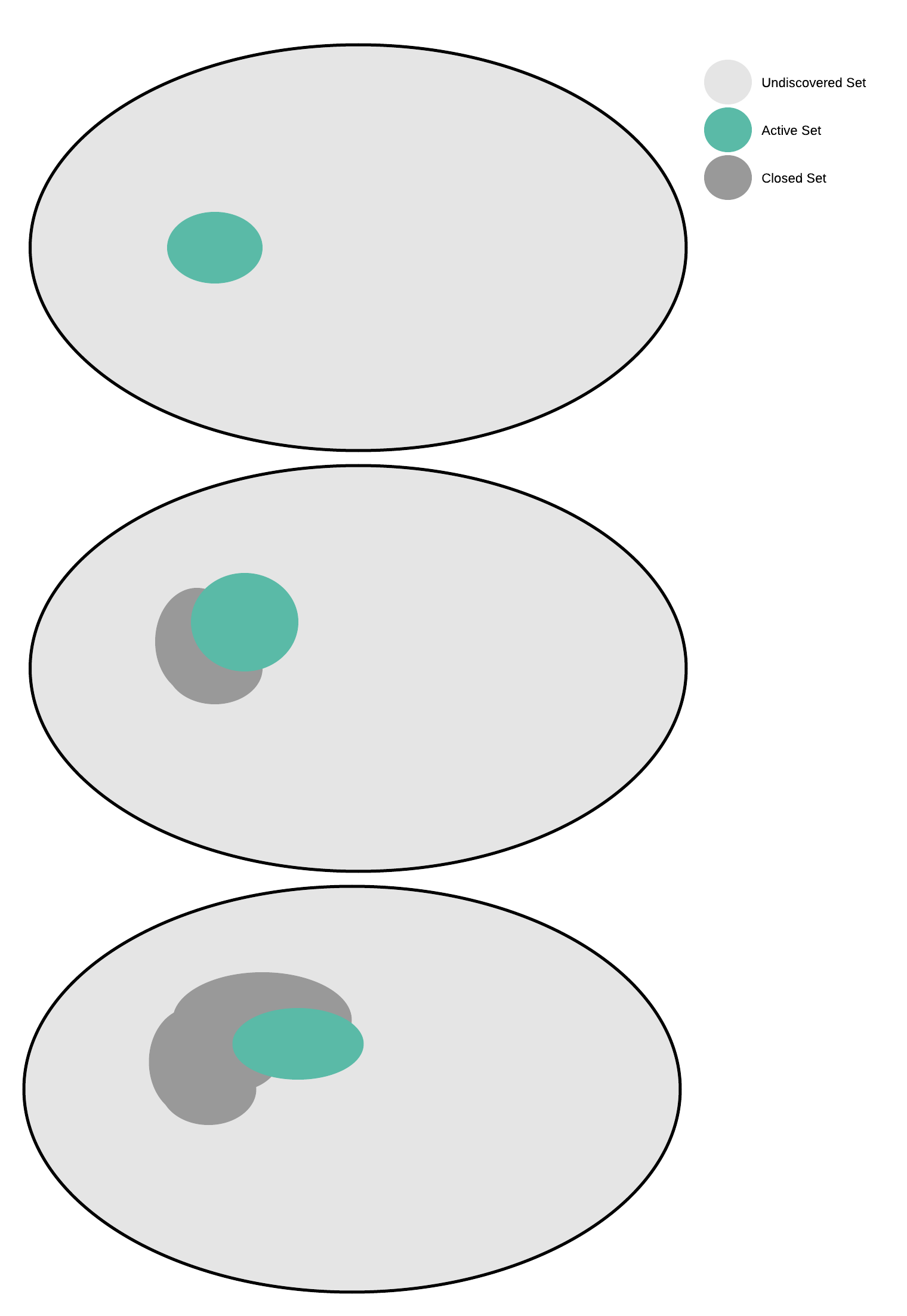
This database of plot points, each with their own conditions, bears some resemblance to Rule Based Systems (RBS). Although, unlike RBS rules, plot points with satisfied conditions are not immediately fired. Satisfied plot points become potentially-accessible in the world but are only fired when accessed by the player in the world.

The DM searches through plot points for those with conditions met. Searching through plot points acts like pathfinding algorithms such as A\* in that there are a few sets of plot points and a state space is being traversed, discovering points and adding them to a subset of points for the DM to evaluate when deciding what to present to the player. This method of ‘discovering’ points is a suitable for reducing the number of points the system needs to consider at any point.

#### *****Navigating the Possibility Space*****

When the application begins, the Possibility Space is initialised and passed to the DM. All points begin in the ‘undiscovered set’ which contains points that have not been seen yet. Points that have no preconditions are then added to an ‘active set’ which describes points that have their logic conditions satisfied. This active set contains the points that can possibly become present in the world. There is also a ‘closed set’ which contains points that are no longer reachable or have already fired.

When the player performs an action in the world, such as accessing a text page on a monitor, the plot point connected to that action is fired. When a point is fired, its effect points are ‘turned off’ and added to the closed set and the fired point itself is then moved to the closed set. The ‘next points’ connected to the effects are checked recursively to see if they should also be turned off. They are also added to the closed set if their preconditions can no longer be satisfied. The next points of the fired point are then checked to see if their conditions have been met and if so, they move to the active set.



**Figure 7 – A visualisation of how the sets can change as points are fired.**

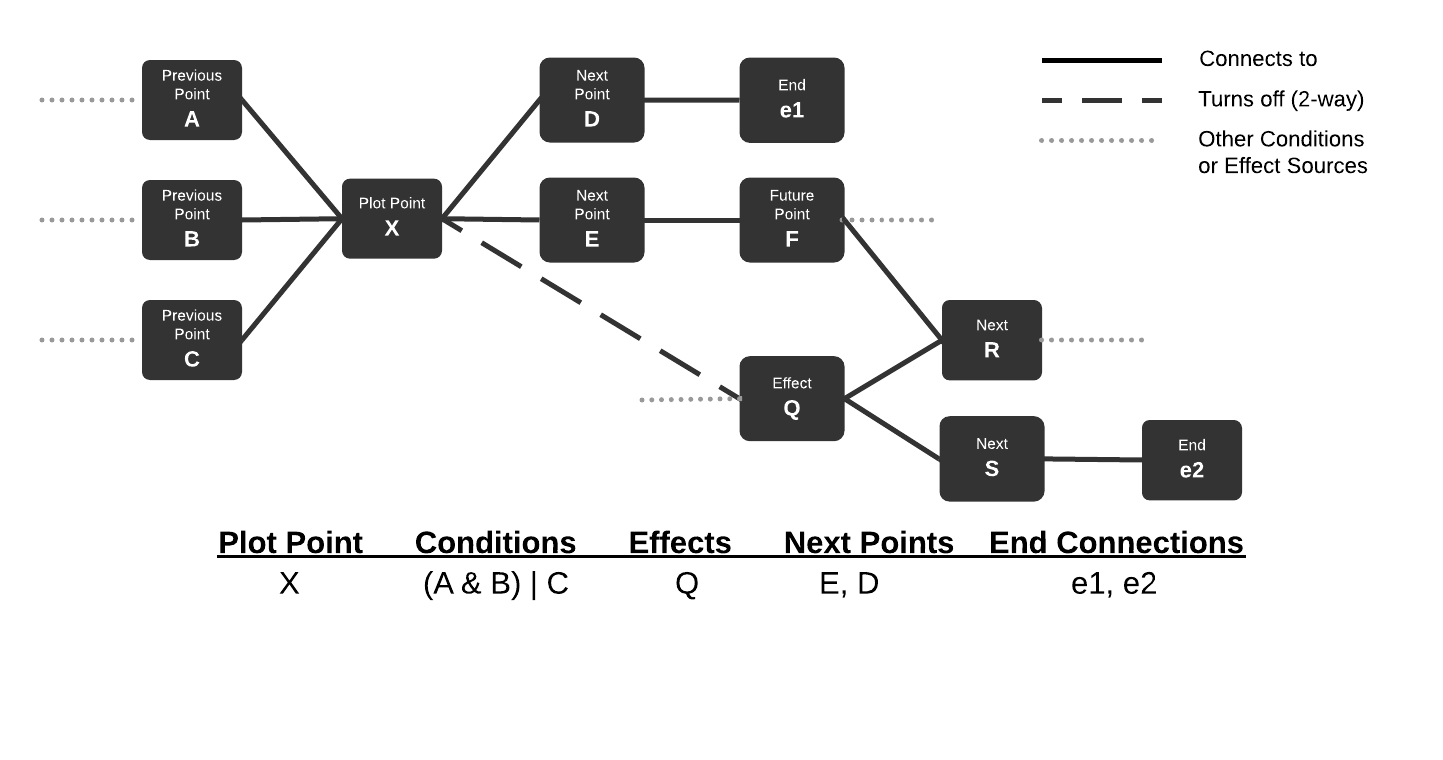
Figure 7 above shows a visualisation of how the sets move and grow within the possibility space as plot points are fired and resolved. The undiscovered set gets smaller with each fired point, the active set shifts both position and size, and the closed set grows as fired points and logically-unreachable points are added to the set.

#### *****Endpoint Dependencies in Graph Structure*****

Later in the system there will be a need to determine if turning off an effect will, in turn, lead to an ‘endpoint’ being turned off. An ‘endpoint’ in this context describes a plot point that marks one of the possible narrative conclusions to the game’s story. There are several possible endpoints and the system must ensure at least one is reachable so that the plot may have an ending.

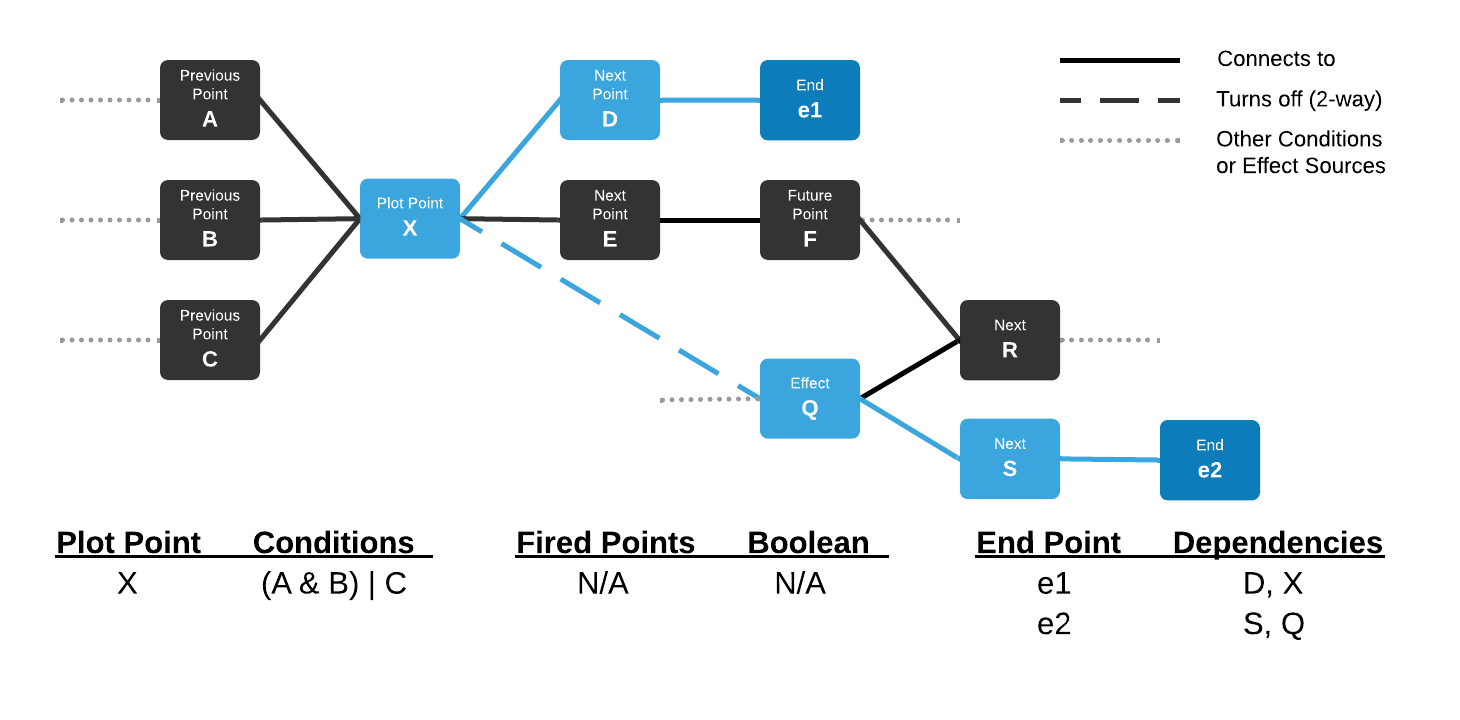
A point has an ‘endpoint connection’ if it exists as part of a branch that leads to one or more endpoints. A point has an ‘endpoint dependency’ if turning off the point will ultimately lead to an endpoint turning off.

Endpoint dependencies are established at initialisation and updated when points are turned off. An endpoint will recursively trace back through its preconditions. During this process each point will check if turning off any of its preconditions would then turn itself off. If turning off a precondition would do so, that precondition then runs its own check and the ‘trace-back’ continues until it can no longer progress.

****

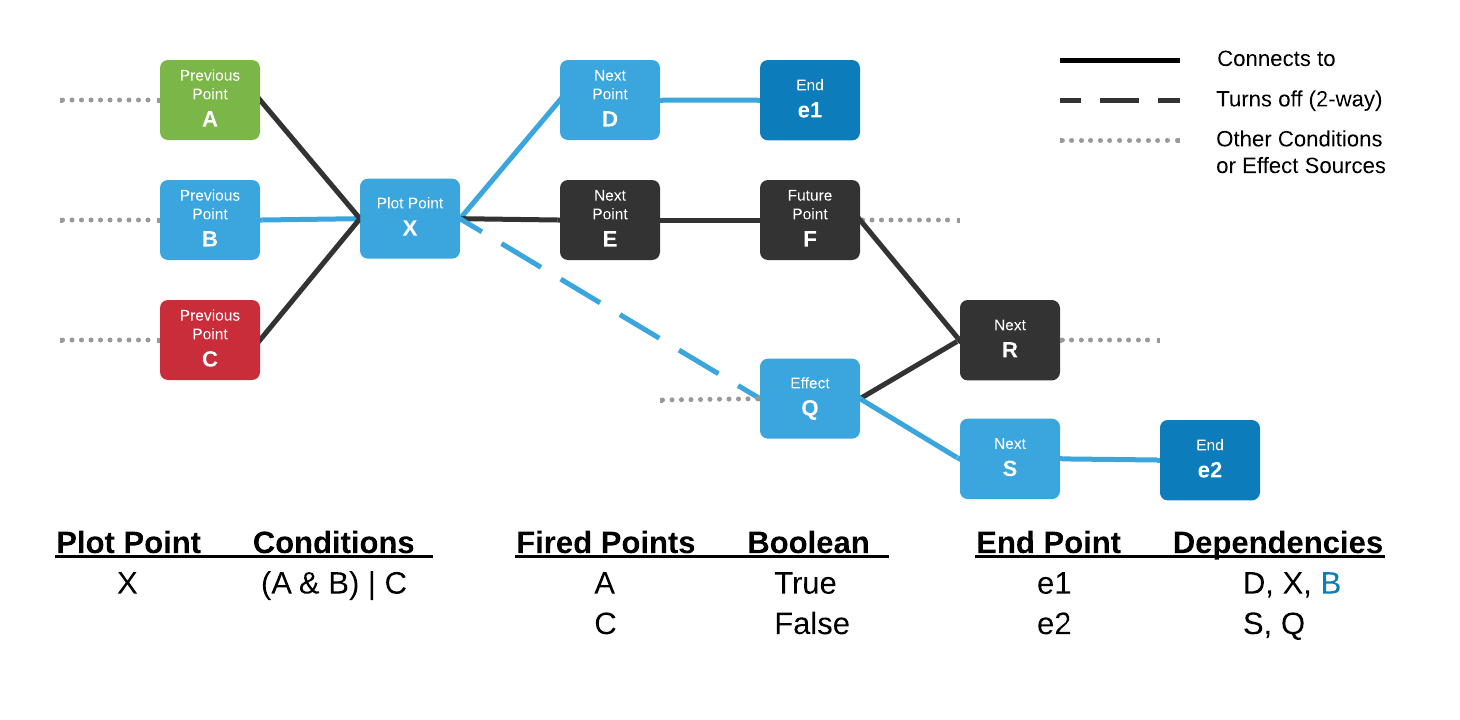
**Figure 8 – Example of a possible graph structure before assigning endpoint dependencies.**

Figure 8 above shows an example of a possible branch structure. Plot Point X has precondition logic using points A, B and C, and an effect Q which is represented by a dotted line. There are two possible endpoints shown, e1 and e2.



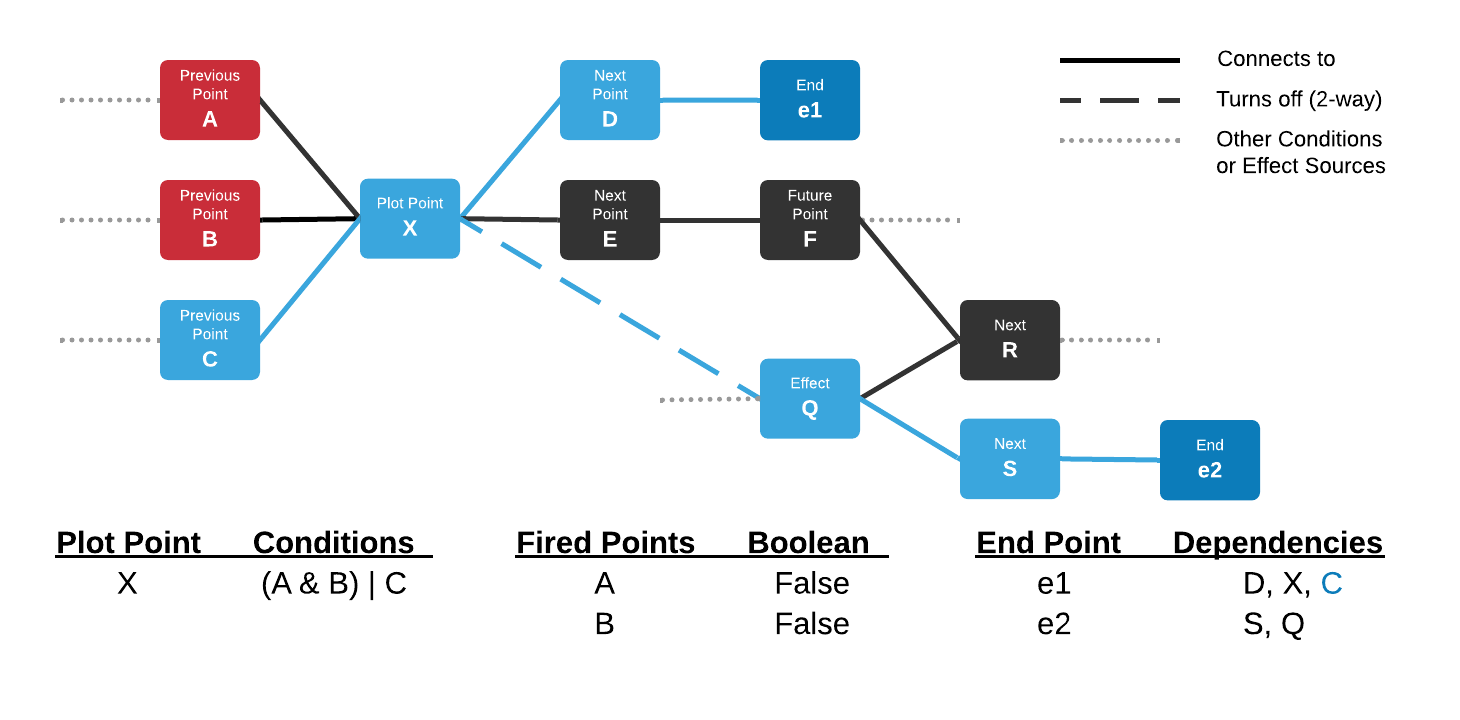
**Figure 9 – Endpoint dependencies calculated during initialisation.**

Figure 9 above shows how endpoint dependencies are initialised. Endpoint e1 traces back to D which is its only precondition and therefore e1 is dependent on D firing. D then traces back to X which also becomes a dependency for e1. The second endpoint e2 similarly traces back to S and then Q. Because X and Q are effects of each other, firing one will turn off the other. Given this configuration, firing X will lead to e1 but makes e2 inaccessible and firing Q does the reverse.



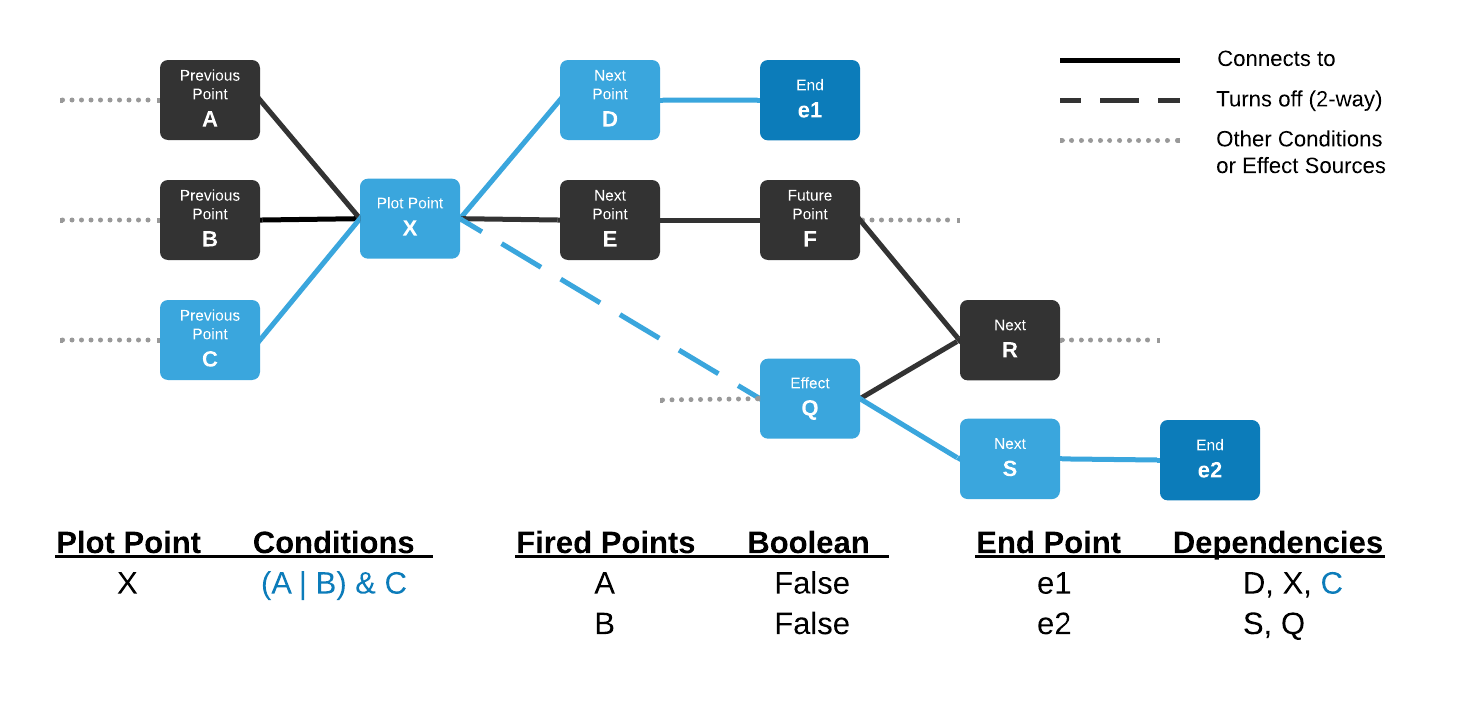
**Figure 10 – Endpoint dependency for e1 gets propagated back to B after A and C are resolved.**

Figure 10 above shows how the endpoint dependencies get updated as points are fired and resolved. In this example, A has been fired true and C has been turned off as an effect. Based on the precondition logic and the points fired, B becomes an endpoint dependency for e1 as it is now required to reach X and so on.



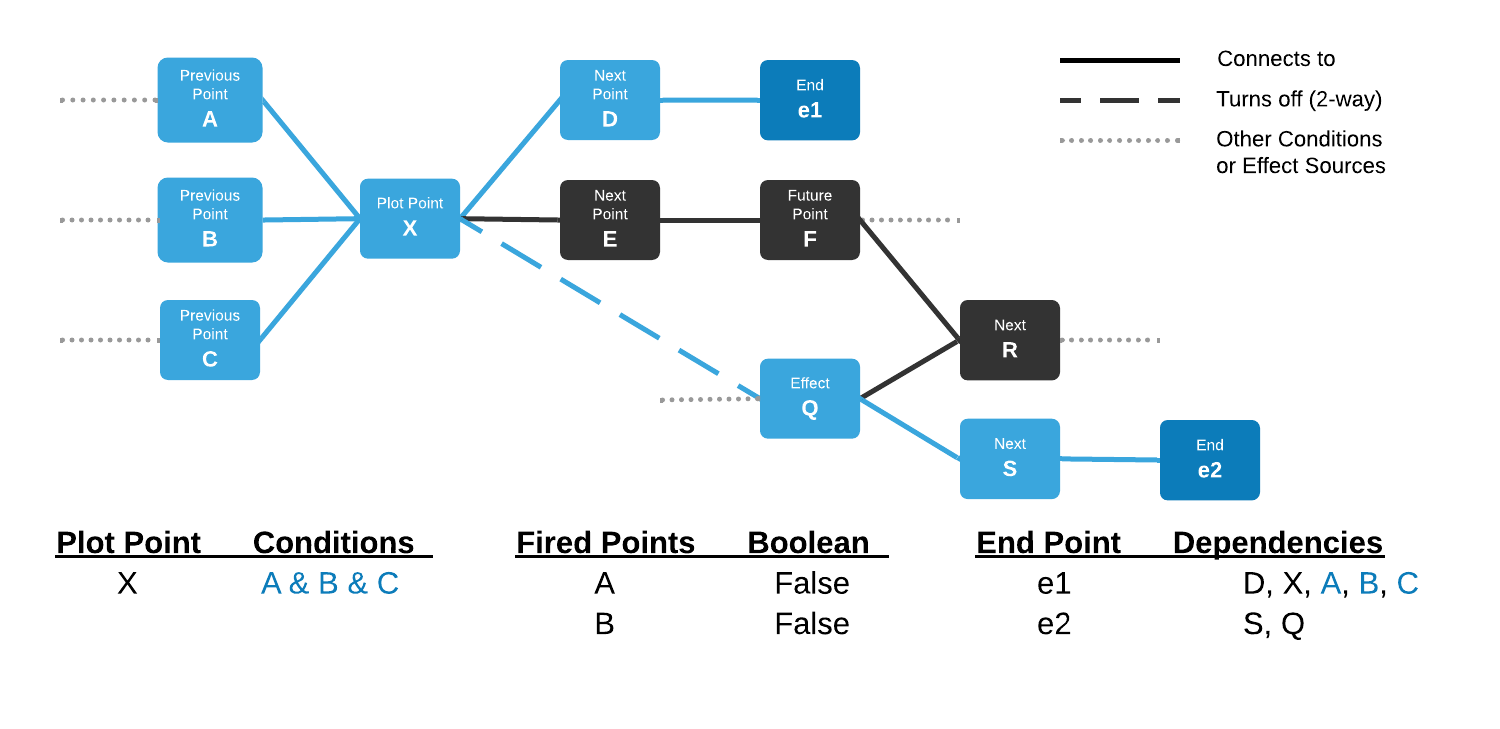
**Figure 11 – Endpoint dependency now assigned to C.**

Figure 11 above is another possible example in which A and B have both been turned off by other effects but C has not yet been fired. With the precondition logic, X would still be reachable if C is fired, and C becomes an endpoint dependency for e1.



**Figure 12 – With different precondition logic, an endpoint dependency exists at C before other points have even fired.**

Figure 12 above shows an example using the same structure but with different precondition logic. If C is turned off by an effect at any point, X becomes inaccessible, even before any points have been accessed.



**Figure 13 – With different precondition logic, multiple preconditions of X are endpoint dependencies of e1.**

Figure 13 above shows that similarly, there may be several preconditions of a point that share endpoint dependencies. In this example, points A, B and C are all endpoint dependencies of e1, and turning any of these points off will make e1 unreachable as a result.

The plot point graph structure and plot point sets constitute the logic of navigating the possibility space and defining what actions are logically viable to make accessible in the game world. However, it is not enough on its own to determine what plot points are preferable to branch the plot. The DM needs to be able to decide what points from the active set to place in the world. For this, the DM constructs and consults a player model.

### 3.3.3 Player Modelling

To decide plot points to make available, plan to prevent undesirable player action, and resolve conflicting events, the DM uses a form of Heuristic Search Planning in conjunction with its logical graph structure. The heuristic for this is based on the player’s predicted playstyle, represented by a set of weights, defined by the designer, which are used to inform a prediction of the plot points they will find preferable. A similarity measurement is calculated between the player model weights and the weights of plot points trying to become available.

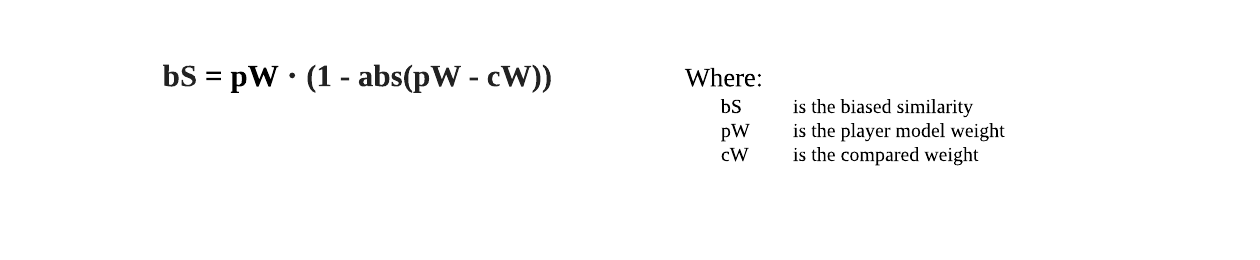
The player begins with a set of weights representing a few different playstyles. Each plot point is annotated with its own set of corresponding weights. When a plot point is fired, the player’s weights and the weights of the connected branch of points is adjusted, reinforcing the future plot points connected to the fired point.

The PM module handles a few calculations that are used in the decision-making. These include calculating the similarity between the player model weights and the weights of a plot point, calculating updated weights for the player model after a point is fired, and calculating an updated set of weights for future plot points.

#### *****Calculating Weight Similarity*****

For the DM to determine plot points to make accessible within the game world, potential plot points are compared to the Player Model through a similarity calculation between the PM weights and the weights of plot points.

The similarity value is weighted, or biased, so that playstyles with a greater value will have more influence over the result. Weights have values between zero and one, but this biased similarity calculation will have a maximum result between these values, where the compared weights are equal to the player weights. The final similarity value is calculated by subtracting the biased similarity of a point’s weights to the PM weights, from the maximum similarity value.

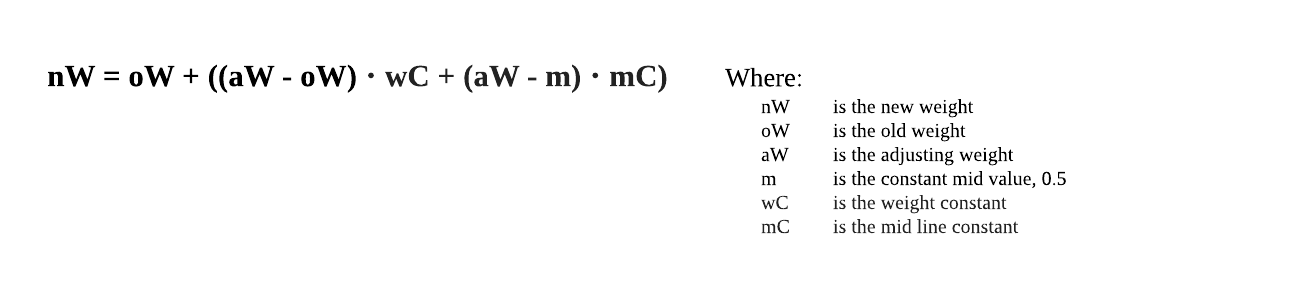


(1)

The biased similarity is calculated from the above equation (1); where bS is the biased similarity, pW is the player model weight, and cW is the compared weight. The result is then subtracted from the maximum similarity to give a final distance heuristic.

#### *****Updating PM Weights*****

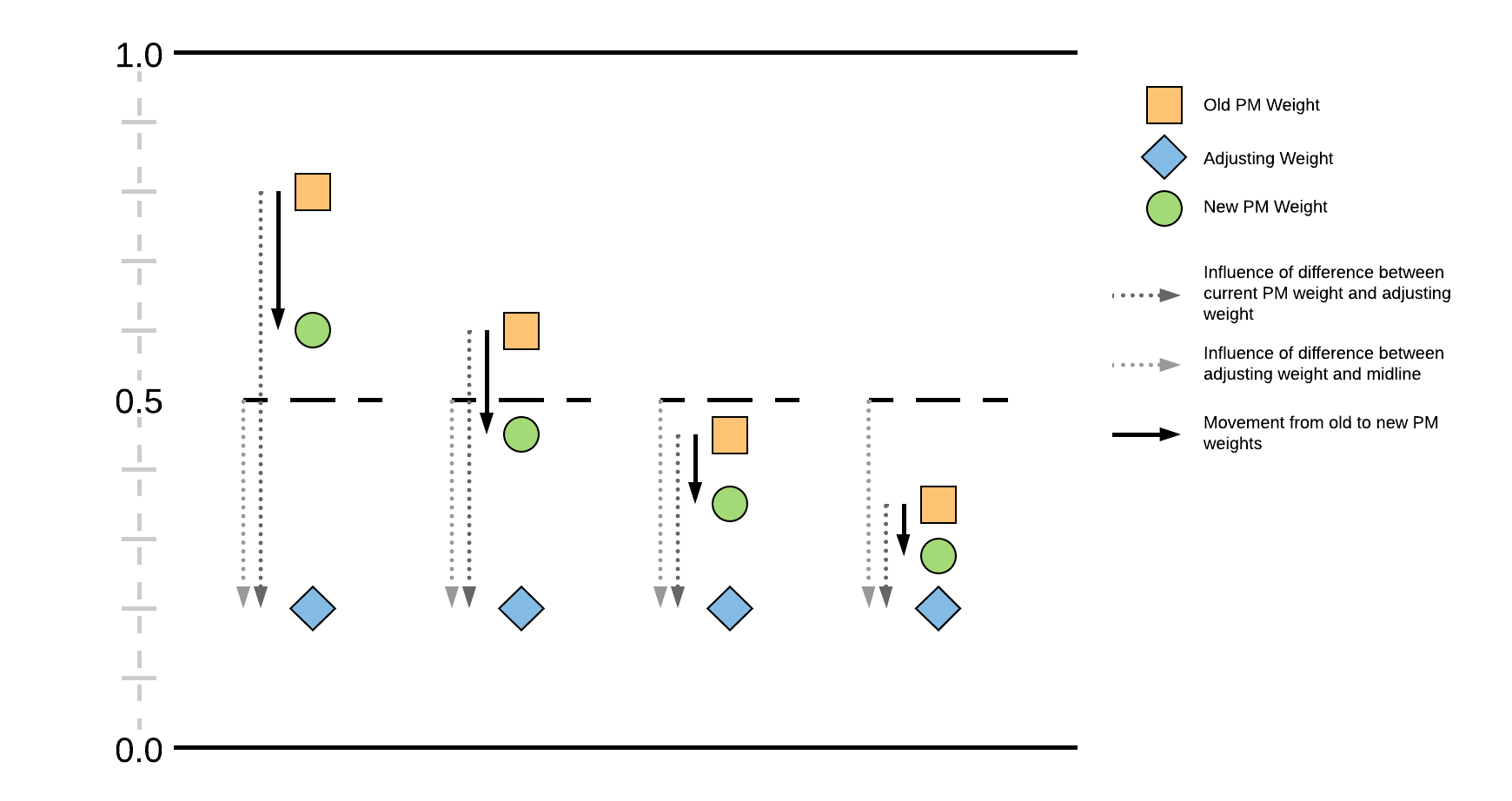
When a plot point is fired, its weights are used to update the Player Model weights. Weights are between 0 and 1, with 0.5 acting as a neutral figure. The aim of this calculation is to reinforce the weights in an incremental fashion with some control over this reinforcement.



(2)

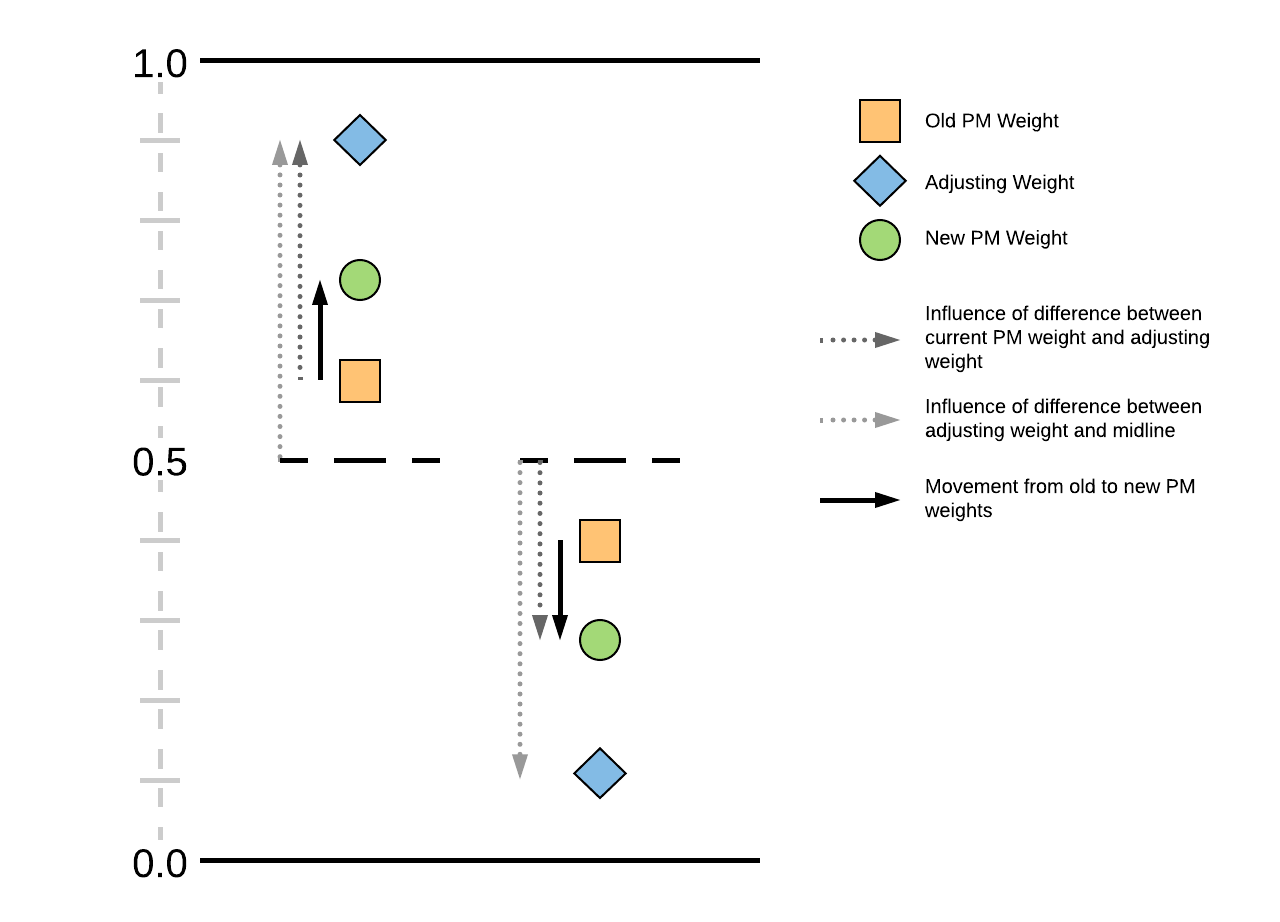
The above equation (2) adds a calculated value to the old PM weight to output an updated weight; where nW is the new weight, oW is the old weight, aW is the adjusting weight, m is the constant mid value of 0.5, wC is a weight constant and mC is the mid line constant.

The increase value is generated using the difference between the old PM weight and the weight of the fired plot point, defined here as the ‘adjusting weight’, the difference between the adjusting weight and the ‘mid value’. There is also a ‘weight constant’ and a ‘mid line constant’ which are used to determine the influence that these differences have on the incremental value. These constants can be tweaked to obtain desired behaviour in the weight update calculation. Below are visualisations of the behaviour of this function within the system followed by example calculations.

****

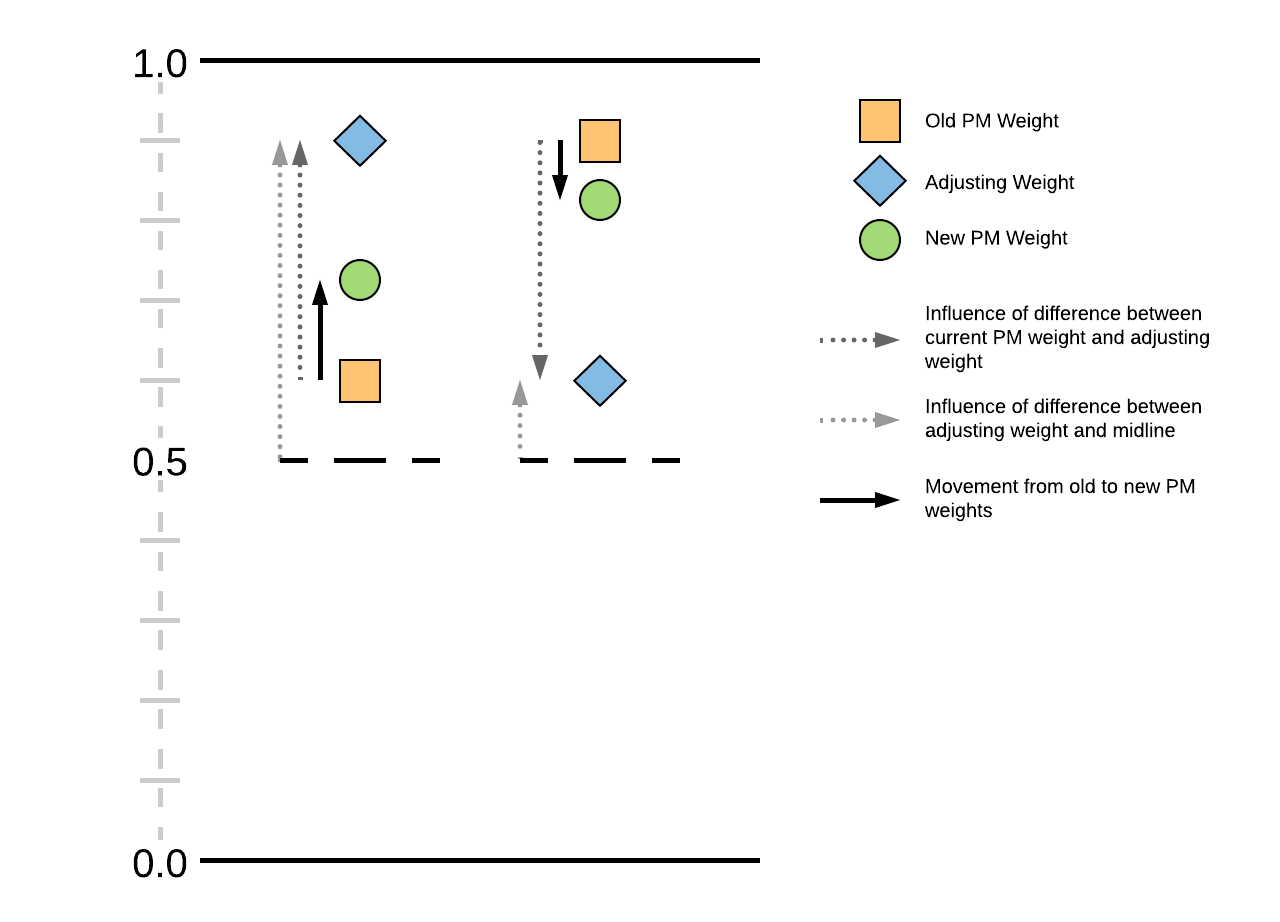
**Figure 14 – PM weight shifting over several steps.**

Figure 14 above shows how a weight can be updated over the course of several fired points. In this example, all fired points have the same weight value of 0.2. The old value moves a significant distance towards the adjusting weight after the first point fires, but this jump gets smaller with subsequent fired points. This is because the PM weight movement is proportional to the distance between the adjusting weight and the old weight. As the weight is moved closer to the adjusting weight, this distance is reduced, thus decreasing the size of the weight change.

****

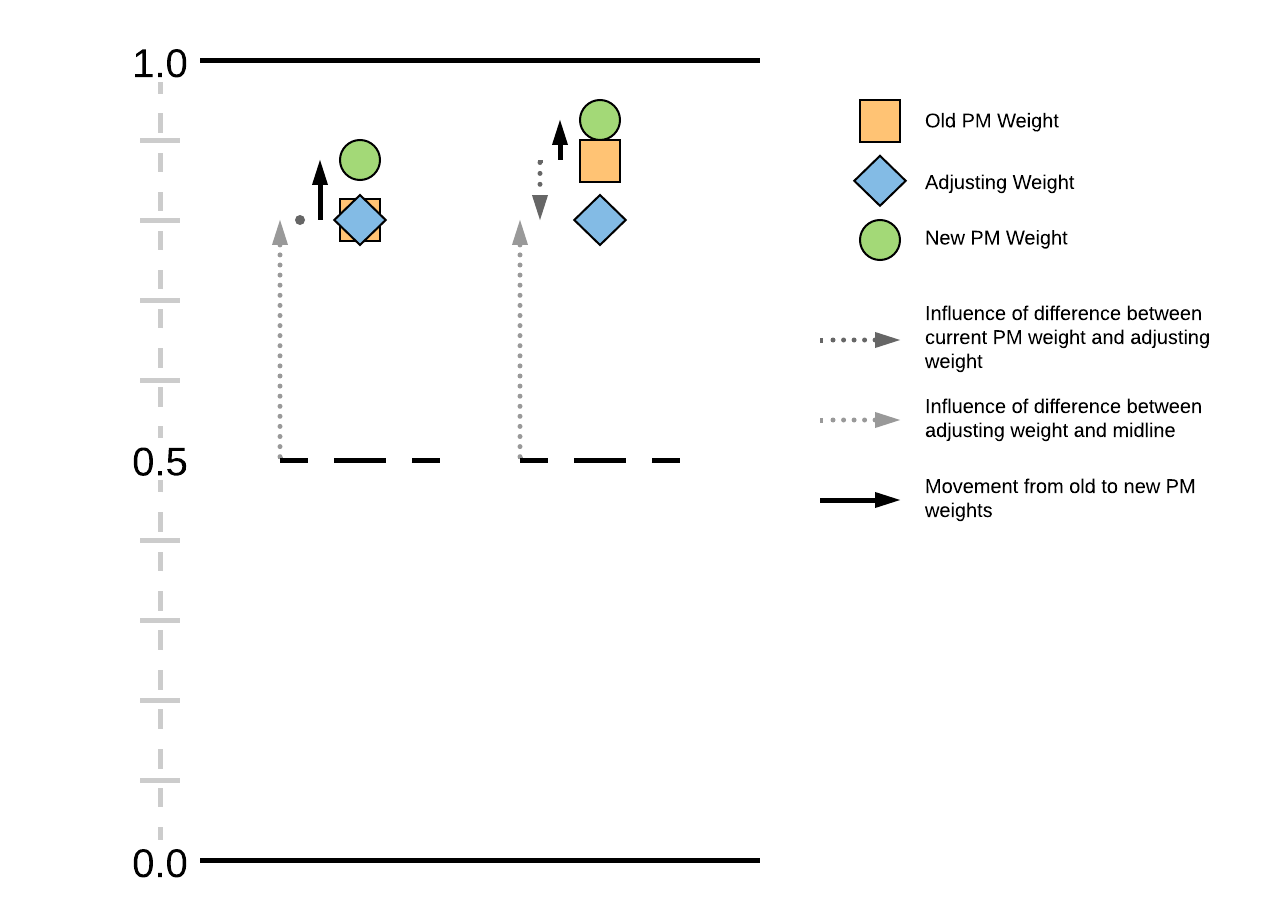
**Figure 15 – Symmetrical weight changes about the mid value.**

This section describes examples of how the function operates. Figure 15 above shows that a weight change could be mirrored about the mid value. The left example shows an increase to a weight and the right example shows an equal decrease to a weight.

****

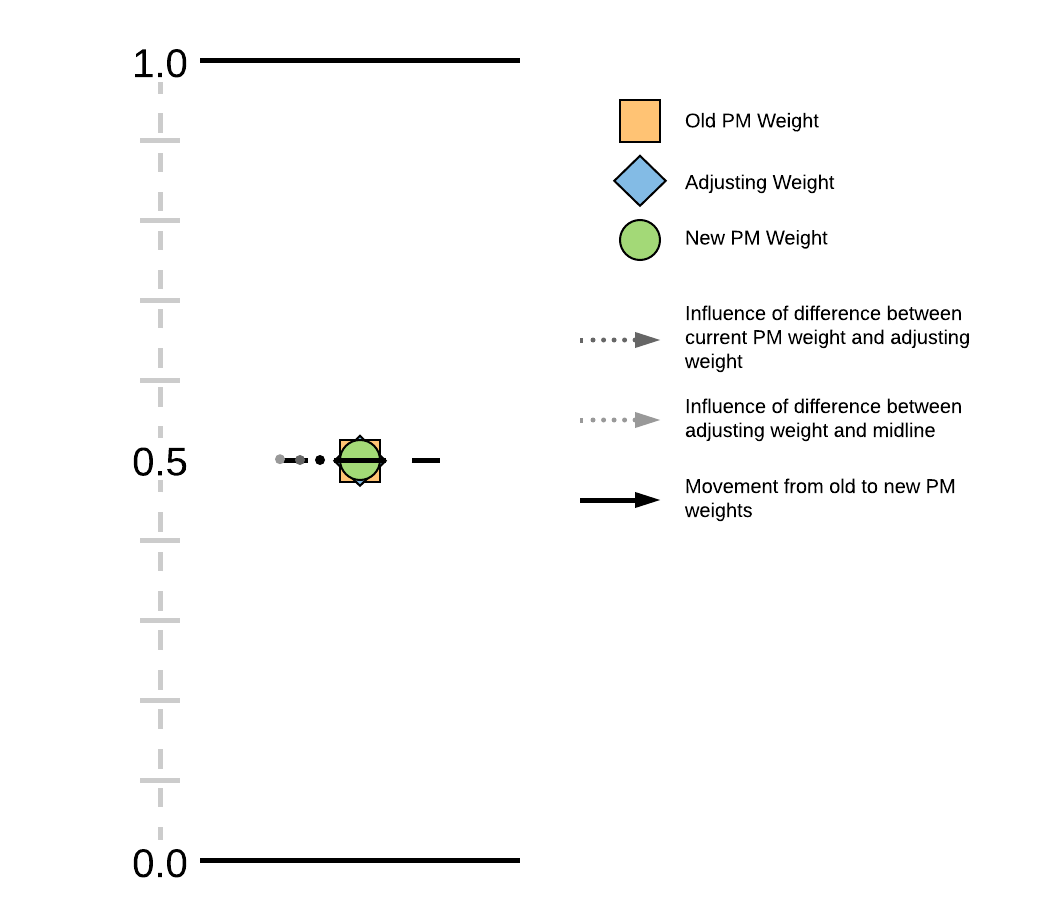
**Figure 16 – Swapping the positions of the old PM weight and the adjusting weight, the decrease in the right example is less than the increase of the left example.**

However, by swapping the weights of the old PM and the adjusting weight on the positive example, the weight does not move an equal distance towards the adjusting weight, as seen in Figure 16 above. The distance between the mid value and the adjusting weight, visualised in the figure by a grey dotted line, is a contributing factor to the weight change calculation.

****

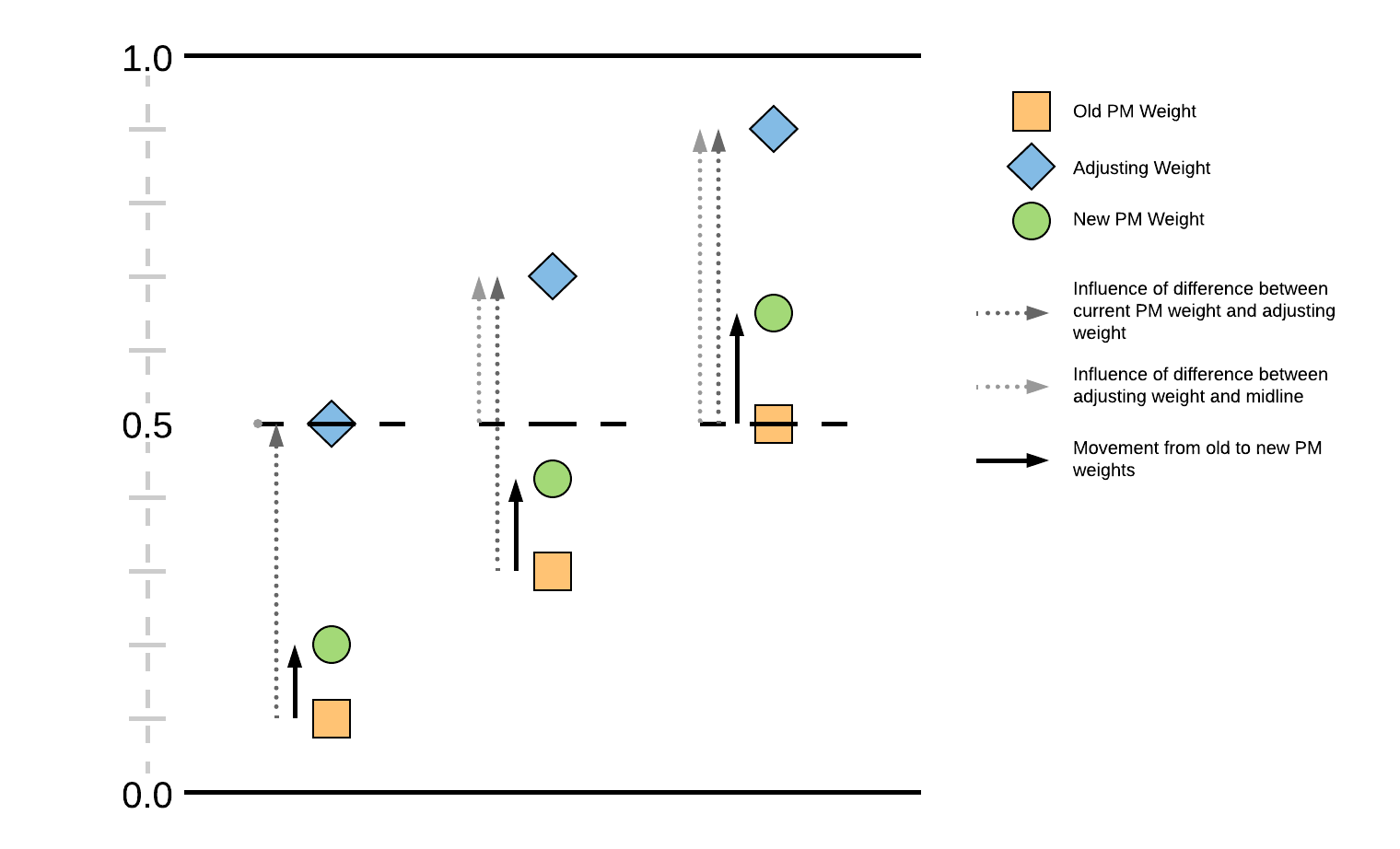
**Figure 17 – PM weights can still be reinforced with adjusting weights equal to or less than the old PM weights.**

Because this distance to the mid value has influence, a weight can also be reinforced slightly even if the adjusting weight is equal to or less than the old PM weight, as seen in Figure 17.

****

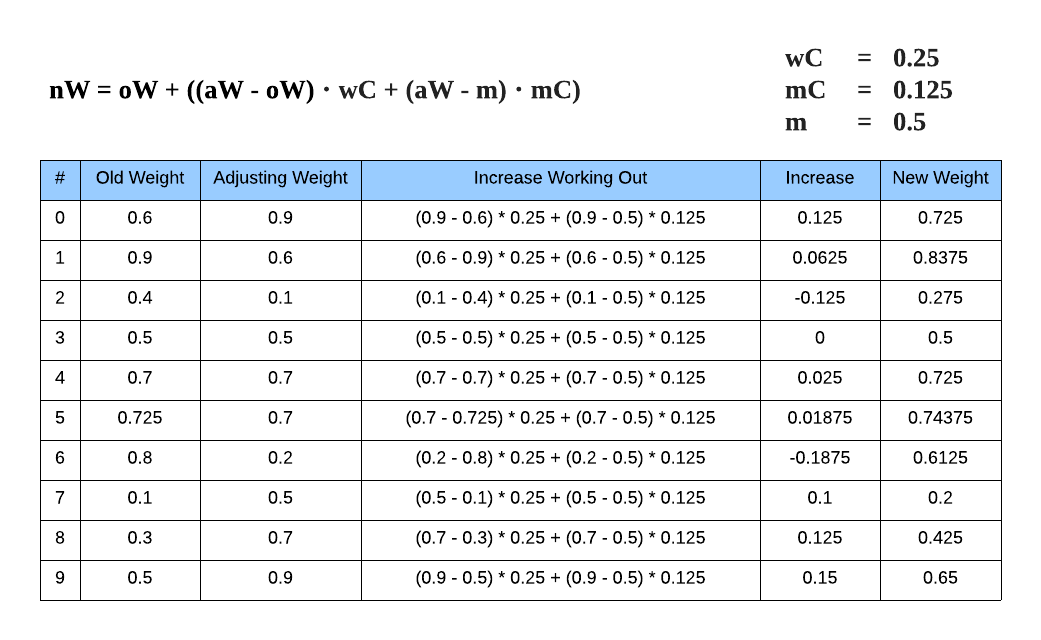
**Figure 18 – No weight change when PM weight and adjusting weight are equal to the mid value.**

The exception to this is in the case that both the old PM weight and the adjusting weight are equal to the mid value, in which case no change is made, as shown in Figure 18.

****

**Figure 19 – PM weight change is affected by the distance between the adjusting weight and the mid value.**

The influence of the distance to the mid value is demonstrated in Figure 19 above, in which the change to the PM weight increases as distance from the adjusting weight to the mid value increases.

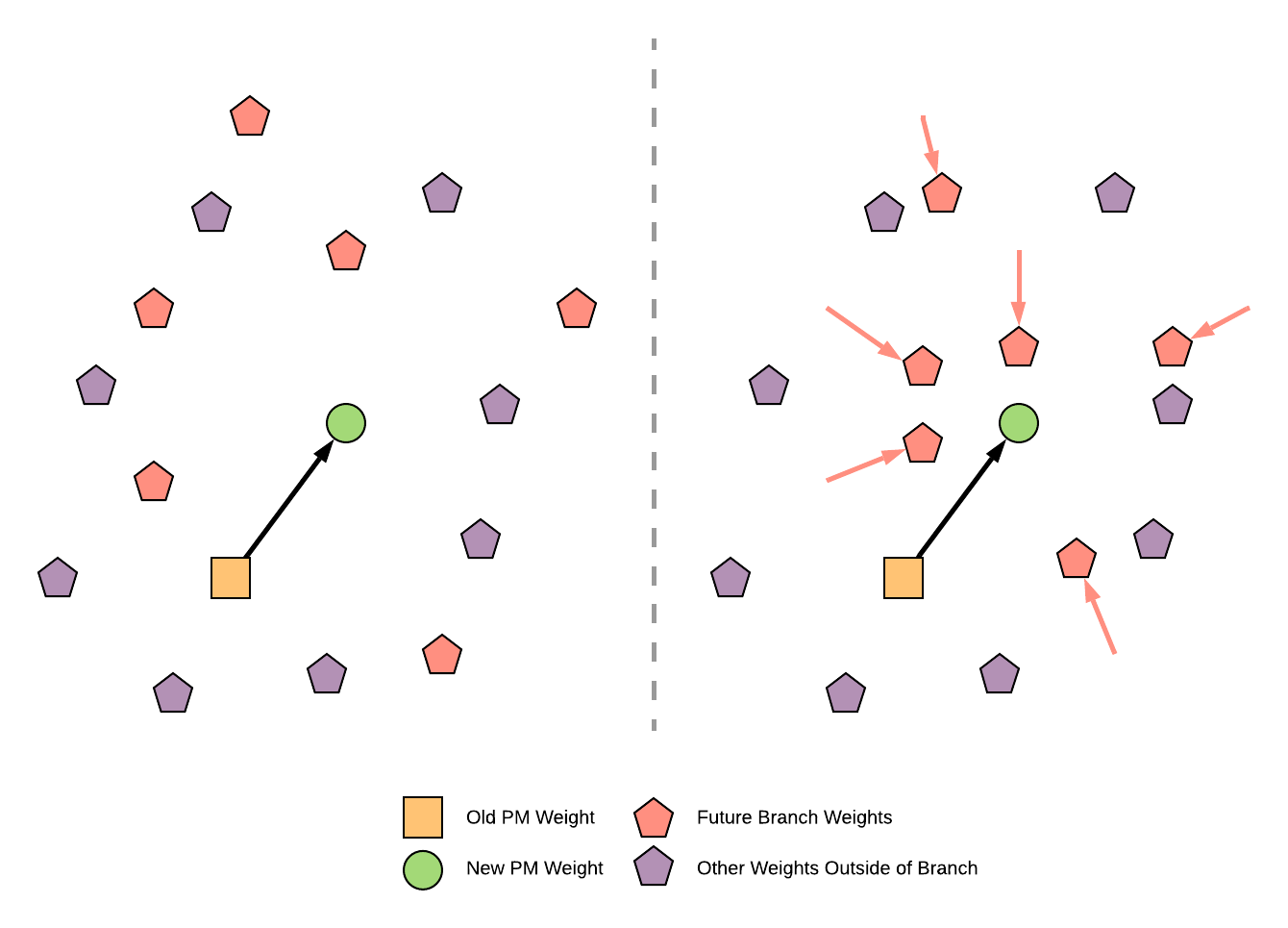
****

**Table 1 – Example calculations demonstrating what has been visualised.**

Table 1 above shows example calculations for many of the situations previously visualised. Rows 0 and 1 correspond to Figure 14. Row 2 corresponds to Figure 13. Row 3 corresponds to Figure 16. Rows 4 and 5 correspond to Figure 15. Row 6 corresponds to the leftmost weight change in Figure 12. The final three rows correspond to Figure 17.

#### *****Updating Weights of Future Points*****

Updated weights are also calculated for future points in a branch so that they are reinforced and become more viable. The future points’ weights are compared to the new PM weights using the biased similarity calculation. This value is then used to linearly interpolate the point weights towards the updated PM weights. This essentially closes the distance between the sets of weights. This is used later when deciding points to place in the world.

****

**Figure 20 – Future branch point weights become more similar to the new PM weight. This can be visualised as clustering around a point.**

Figure 20 above shows a visualisation of how weights of future plot points in the same branch as the fired point converge towards the updated player weights. The points with weights farther away move a small amount towards the new PM weights and the closer points move a larger amount; their movement is proportional to their similarity to the PM weights. This movement then improves the similarity value of future branch points, which makes them more likely to be put in the world.

These calculations that update the player model, update future weights, and calculate a similarity value are all used during ‘information replacement’ discussed later in the chapter.

### 3.3.4 Player Action

For plot points to be fired by the DM, and the world to be updated with new plot points, the system modules need a tangible connection to the game world.

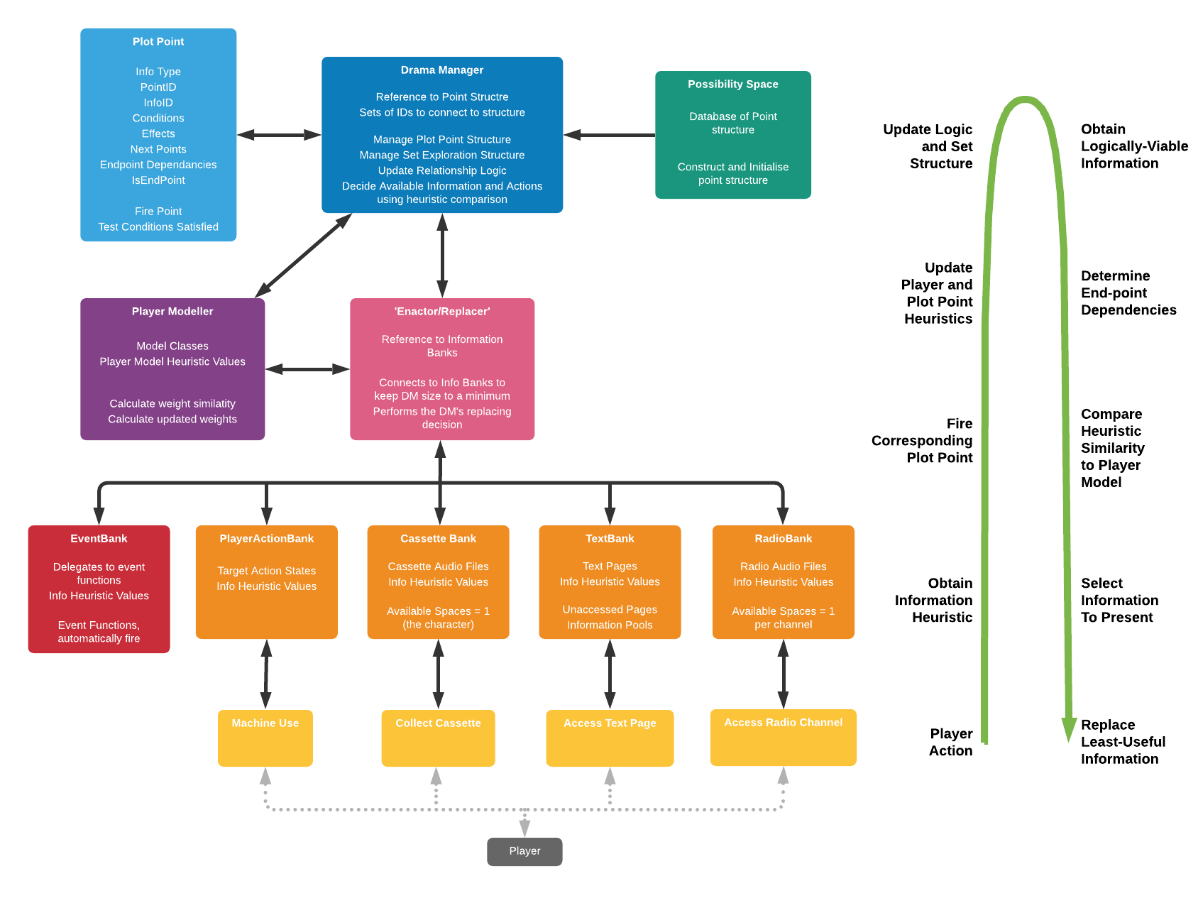
The DM does not have a direct connection to the world but acts through another module called the ‘Information Enactor’ or ‘Information Replacer’. The purpose of this module is to pass plot point information up and down the system, and to perform the information replacement process after a plot point is fired. The Enactor was made separate to the DM module but can be considered part of the Drama Manager IA as the information replacement task is one of the most important functions of the DM.

Plot points defined in the Possibility Space graph connect to one of several Information Banks which contain the associated narrative information and their set of weight values for the Player Modeller to manipulate.

When the player takes an action, such as accessing a monitor’s pages in a room, the action is connected to the index of a plot point. This is passed to the DM through the Enactor, where the plot point is fired, and the DM navigates the state space. The plot point’s weights are then obtained from an Information Bank and the Player Modeller performs its updates. Finally, the DM calls the Enactor to perform plot point replacement in the world. To the player, the narrative information connected to the action they took is presented, for example, through text.

### 3.3.5 Information Replacement

Information Replacement is done as the last task of the system after a player takes an action. The whole system operates in a ‘turn-based’ fashion, in which the player takes an action, the system performs internal updating and reasoning, before outputting changes to the world.

****

**Figure 21 – Diagram of how modules connect to each other****, to the world actions, and finally to the player. On the right is an arrow that shows the processes that occur from player action to information replacement.**

The overall system architecture is shown in Figure 21 above. This is a more complex version of Figure 4. It shows the processes of the system and how information is passed up and down the system structure. As before, the player takes an action through the world’s interaction methods. The action has an associated plot point reference. This reference is passed up through the Enactor to the DM where the point is fired. The DM navigates the space, the PM updates, and the Enactor begins the replacement process.

Information Replacement is the main feature of the Drama Management System. During this process, the IA evaluates what plot points are logically viable, and then must decide which are preferred to put in the world. The IA puts the most preferable plot points in the world while removing the least preferable points.

The logical viability is obtained through the ‘set logic’ in the DM, as the state space is navigated. The non-closed plot points with satisfied precondition logic are contained within the active set. The Player Modeller’s weight similarity calculation is used as a heuristic for determining preference.

Within the Active Set, there are plot points that are already present in the game world, called the ‘W-set’ or ‘World Set’, and points that are not currently in the world, called the ‘!W-set’ or the ‘Void Set’.

The premise of replacement is that some of the ‘!W-points’ may be preferable to the player based on their predicted model. Replacement allows for the most appropriate ‘!W-points’ points to be made available, while removing points from the ‘W-set’ that have either been closed or are otherwise less viable.

Replacement has several considerations and constraints that must be handled, while granting flexibility for a scalable possibility space with complex requirements. Information placed in the world may only make sense to be found in certain locations. Information already in the world also has this constraint and if replaced, could perhaps still fit somewhere else. Plot points not currently viable may also become viable in the future as the player model changes, so unviable points cannot simply be closed.

Some points lead to an endpoint whereas others do not. This factor leads to several additional considerations. It is undesirable for the system to present a less-preferable endpoint as an option to the player based on the player model, thus endpoints must be evaluated for similarity. There should be a range of acceptable endpoints to be available to the player. Endpoints outside this range should not be made accessible to the player, and the points that lead to undesirable endpoints must not be present in the world. Finally, points that turn off a viable endpoint, through the endpoint dependency structure, should not be accessible unless they themselves are a dependency for another viable endpoint. The process of determining the viable range of endpoints and disallowing unviable endpoints is called Endpoint Conflict Resolution.

The replacement process begins by updating the W-set and !W-set after the DM has updated the Active Set and marking newly-closed W-set points to be replaced. Endpoint conflict resolution is then conducted, again marking points as unviable so that they are either replaced or cannot be used to replace.

After the resolution, all active plot points, from both the W-set and !W-set are ordered by viability into a list and given a ‘viability index’ of that list, which will be used to determine which points are more viable, and thus preferable to have in the world. This sorting is done by three factors, the similarity of the points’ weights to the PM weights, whether the point eventually leads to an endpoint i.e. if it has an endpoint connection, and whether an endpoint dependency exists out of the list of endpoint connections of the point. Points without an endpoint dependency or connection are ordered into the list by their similarity to the player model, points with a connection are ordered in a separate list before being appended to the end of the viability list. This is also repeated for points with an endpoint dependency. Points marked as unviable are put at the very front of this list so that they are given a low ‘viability index’. In this way, the most replaceable points are at the beginning of the list, followed by points that do not lead to an endpoint, then by points that do lead to an ending, and finally by points that a viable ending depends on.

Actions in the world have an ‘element space’ which ‘houses’ a reference to a plot point. An element space also contains a list of all !W-set points that can replace the ‘housed point’. This is called the ‘competition list’. Because the housed point could go on to replace another point in the W-set that is less viable, the element space contains a function that returns the least viable point in the world that would be ejected because of the housed point being replaced i.e. the housed point would replace another point which may replace another and so on until no more replacement could be done. The least viable point found by this process will be referred to as the lowest trace point’. A point from the !W-set is added to an element space’s competition list if it’s viability index is greater than the index of this ‘lowest trace point’ and if the point lists the space as one of the places it could be found.

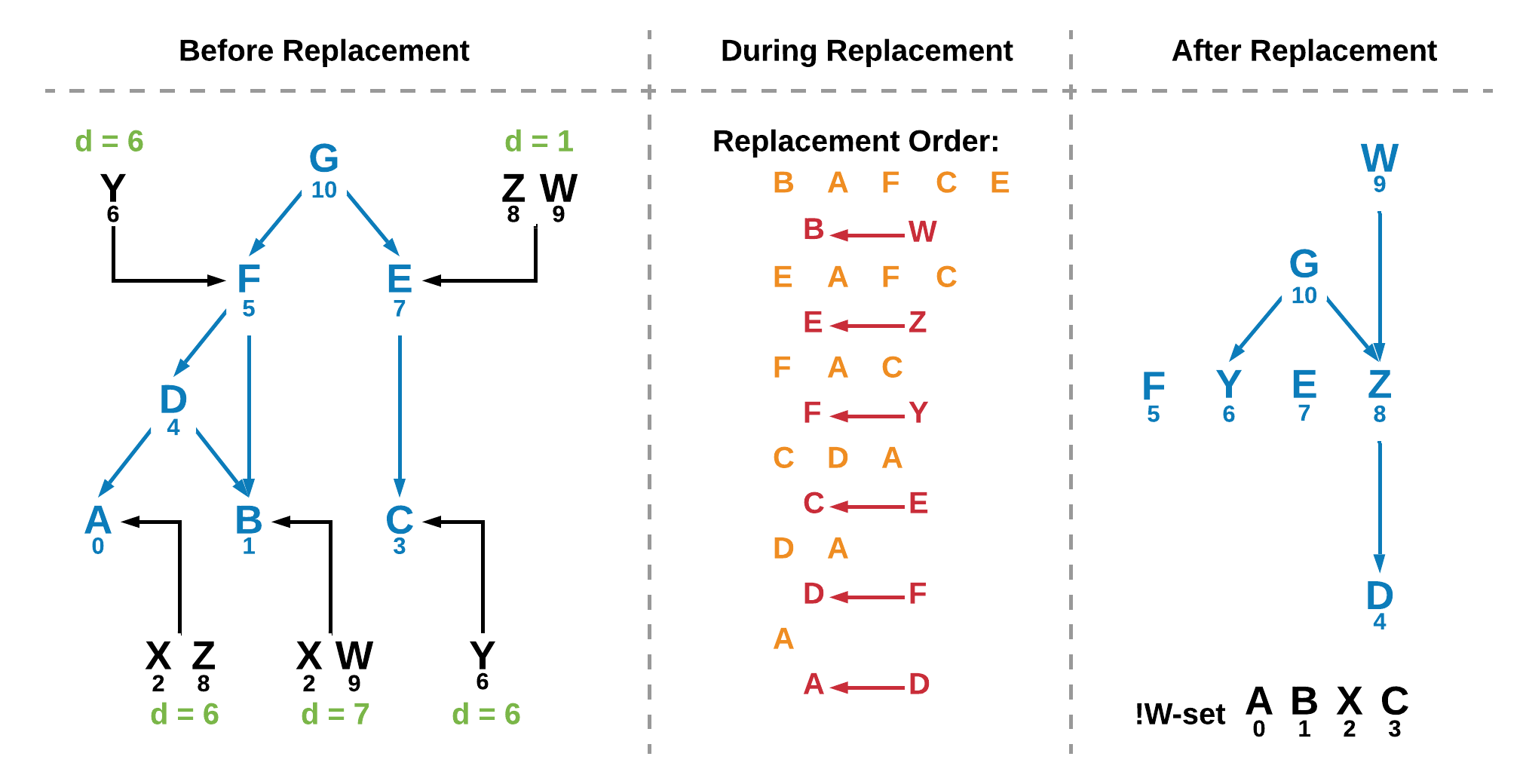
After points are given their viability index, the competition lists of every un-accessed space are established and the spaces are ordered by greatest distance in viability index between their best two competitors, if that many exist, and sub-ordered by ‘lowest trace point’. This sorting determines the order points are replaced. If a space only has one competitor, its ‘competition distance’ is equal to the index of the sole competitor.

The space with the greatest competition distance is replaced first. The best competitor from its list replaces the housed point. The housed point is removed from the W-set and added to the !W-set and it is added to the competition list of spaces it could appear in if it beats their lowest trace point. The winning competitor becomes the new housed point of the space and is removed from the competition lists of its possible spaces. The sorting of the spaces is then updated and a new space with greatest competition is found. This process is repeated until no more replacements can be made.

To summarise, replacement is done in the following steps:

* Update W-set and !W-set
* Resolve endpoint conflicts
* Sort points by viability
* Establish competing points for each element space
* Sort order of replacement by difference in best competitors followed by lowest trace point
* Perform replacement on the space with greatest difference in best competitors, replace with best competitor
* Update sets and element space competitors and re-order replacement
* Repeat until no more replacement can be done

This section finishes with an example situation, with the world set represented as a tree structure, with incoming void set points.

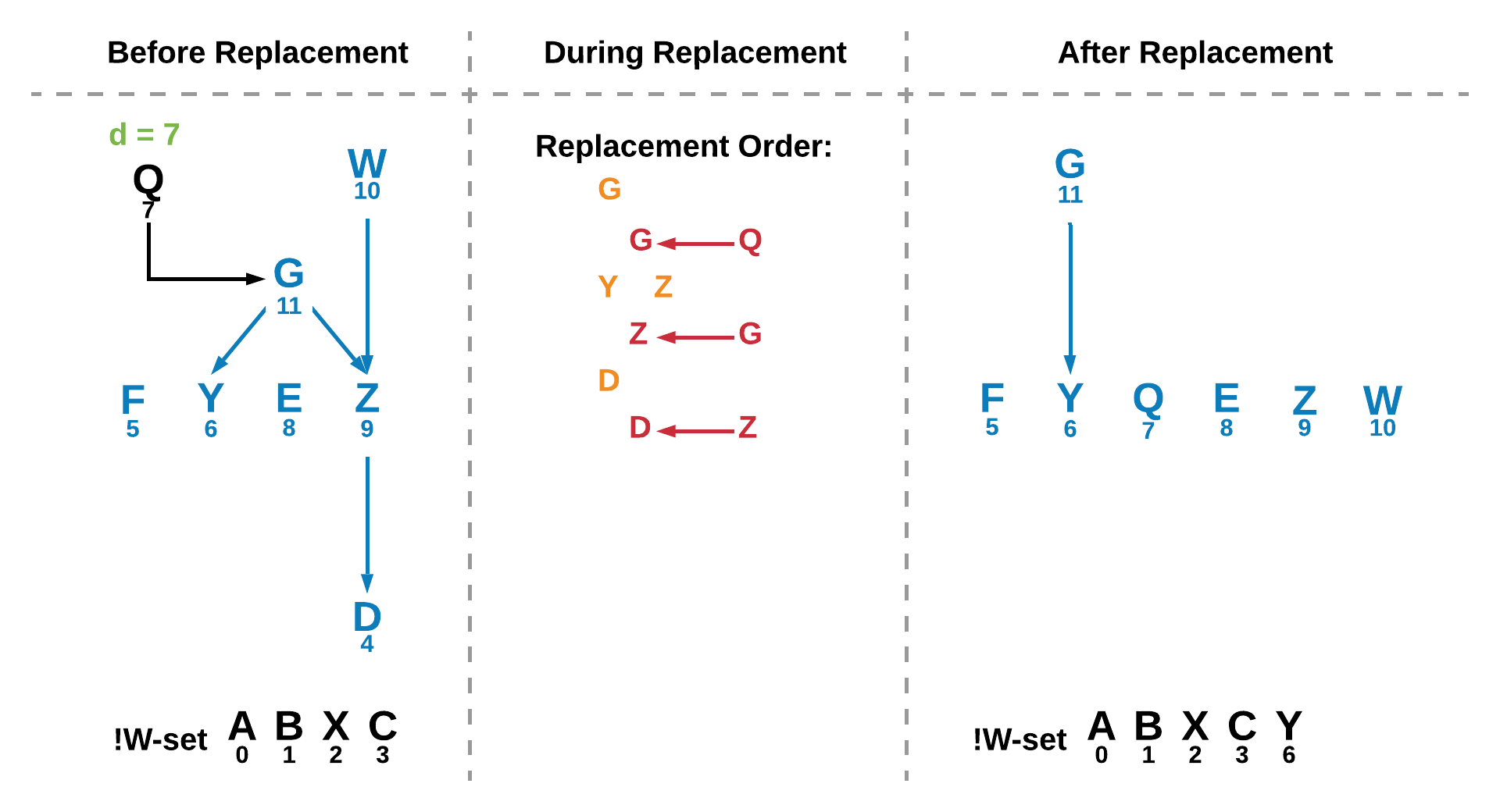


**Figure 22 – Example of a round of replacement****, where the world set is represented by a tree structure. The left tree structure becomes the right tree structure at the end of replacement.**

In Figure 22 above, the left tree represents the points in the W-set and how they could replace other points if they are themselves replaced. The middle column shows the replacement order and steps through it. Once replacement is finished the tree becomes that shown on the right.

The values under the letters represent the viability index of competing points. The green values shown in the left tree represent the difference in viability index between the top two competing points. If only one competitor exists, the difference value is equal to the index of the lone competitor.

After each step of replacement, shown in the middle column, the competition lists are updated and new differences in viability indices are calculated. The list of spaces is reordered and the element space with the greatest difference value is replaced by its best competing point. This process is repeated until no more replacement can be done.



**Figure 23 – Example of another round of replacing** **in which point Q is less viable than G but more viable than the lowest trace point D.**

In Figure 23 above, during another round of replacing, point Q has a viability index less than that of the housed point it is trying to replace but greater than D, the lowest trace point. In short, Q is less preferable than G, but replacing G will eject D, which is less preferable than Q. In this case, Q replaces G which replaces Z which then replaces D. The tree now becomes that shown on the right of the figure.

## 3.4 System Summary

The system can be summarised as a series of steps:

* A possibility space defines an acyclic directed graph of plot points.
* Plot points have precondition logic and effects.
* Plot points with satisfied logic are moved into the active set.
* When an action is taken by the player, a plot point is fired.
* When a plot point fires, its effects are turned off and next points are checked to see if their condition logic has been satisfied, the DM then updates its sets.
* The PM uses the weights of the fired plot point to update the player model.
* The PM then updates the future plot points in the branch using their similarity to the updated player model.
* The Enactor uses the active set to perform information replacement; the less-viable and closed points are removed from the world and the most-viable points are put into the world.

## 3.5 Other Considerations

This section briefly discusses additional considerations and constraints of the system.

Due to the nature of trajectory planning, the computational complexity of the system is rather high, which can prove a challenge in a real-time environment where attributes such as framerate are important to the enjoyment of the experience.

User Experience (UX) and Human-Computer Interaction (HCI) are also considerations of the project as the design of these attributes have influence over how the player interacts with the world. Although UX design and narrative design are separate areas of the prototype, poor design of UX can negatively impact the player’s enjoyment and in turn skew their perspective of the plot itself. A bad user experience can hinder the narrative experience.

There are some final constraints added to the system. These features are in place to reduce the search size for replacement and improve the integrity of the story structure by removing redundancies. When accessing a text page on a monitor, the monitor would return the plot point in the room with the closest similarity to the Player Model, i.e. the most viable point. Additionally, when a plot point is closed, its preconditions are checked to see if they are no longer necessary. If they lead to no other undiscovered points, they too are closed, and have their conditions checked. Finally, when an action is taken, the location it is found in, within the monitor, is no longer available for plot points to be placed in during the replacement process. When a plot point from the active or undiscovered set runs out of possible locations to be accessed from, it is closed. The preconditions and next points in the point’s branch are then checked to see if they too should close.

## 3.6 Data Collection and Analysis

Data collection is done through trace testing. The program records the history of the plot points fired and associated information, discussed in the next chapter.

Trace testing is done by the designer, where they go through possible playthroughs of the game, recording the history of actions, PM weights and endpoint reached, while also noting discrepancies or providing feedback. During this testing, the designer also makes qualitative notes on the narrative experience.

The results are evaluated to determine how the system affects perceptions of the plot and enjoyment, including how much impact player actions appear to have and how logical the plot structure appears to be. The results are also used to suggest improvements to the system in terms of both computational improvements and design improvements.

The dataset created for testing consists of 6 endpoints, with branches that are weighted to reinforce the player model towards an endpoint. There are four main branches, with two that sub-branch to different, yet related endings. There are four weight values per plot point, with different combinations of higher-value weights that reinforce the PM in the direction of connected endpoints. The dataset is also designed so that it contains branches that have a great amount of direction for the player, and branches that encourage a broader exploration of the environment to follow a plotline.

# 4 Results

This chapter begins establishing requirements for testing the system, followed by a description of the dataset created, before discussing the testing process and results obtained.

## 4.1 Requirements and Dataset

The requirements of the testing are as follows:

* The dataset must be of a substantial size to represent a complex narrative environment.
* The directed graph represented in the possibility space must lead to multiple different endpoints.
* There must be one or more branches that lead to different endings that can be found in the same location, in order to determine if the system will choose to present the preferable endpoint.
* The dataset should be written with careful logic, preventing circular dependencies and preventing a point in the branch from ‘turning off’ its future next points i.e. a point that leads to an endpoint should not turn off that endpoint.
* The dataset should be written with consideration of performance; a plot point that is widely available to a broad range of possible actions/locations will require more searching during replacement. The solution must run in real-time.

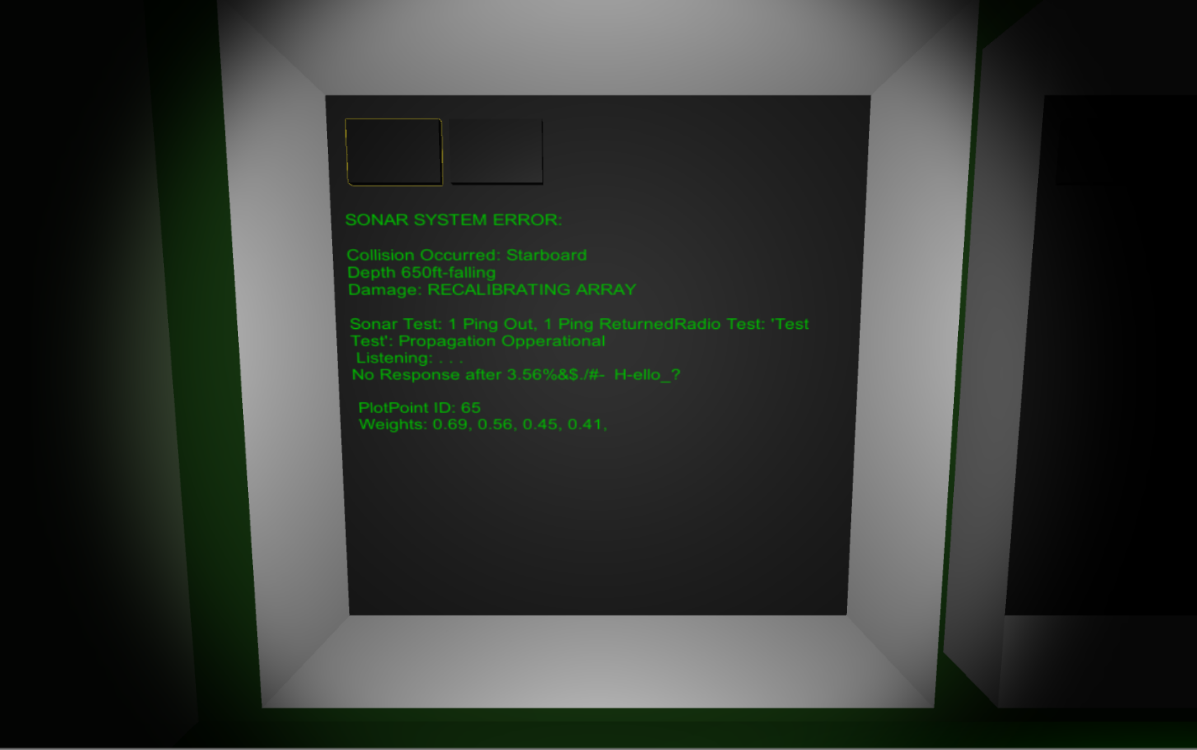
The game environment has 32 monitors in total, with 2 pages of text each, where accessing a page is an action that is connected to a plot point. Thus the ‘world set’ or ‘W-Set’, as described in the previous chapter, begins with a size of 64 and decreases in size as the player takes actions. The possibility space created has a size of 150 plot points, with 64 placed in the world at the beginning of the game.

The plot points were graphed by hand, with each point being given preconditions, effects, weights, locations to be accessed from, narrative text, and whether the point is an endpoint. The raw dataset can be found in Appendix A.

Testing was done on a computer with an Intel(R) Core(TM) i7-6700 CPU @3.40GHz with 32GB RAM and a NVIDIA GeForce GTX 1070 graphics card.

During testing, it was found that the ‘trace down’ recursive algorithm to find the lease viable point that would be replaced was too expensive, slowing the game out of real-time when certain actions were taken. The algorithm was adjusted to improve the speed to an acceptable level without a severe drop in optimality. The trade-off between speed and optimality is discussed in the next chapter.

The prototype displays information for the tester to observe changes made with each action taken. Figure 24 below shows an example player action. Narrative text is displayed, followed by the plot point ID and the associated weights when the point is fired.

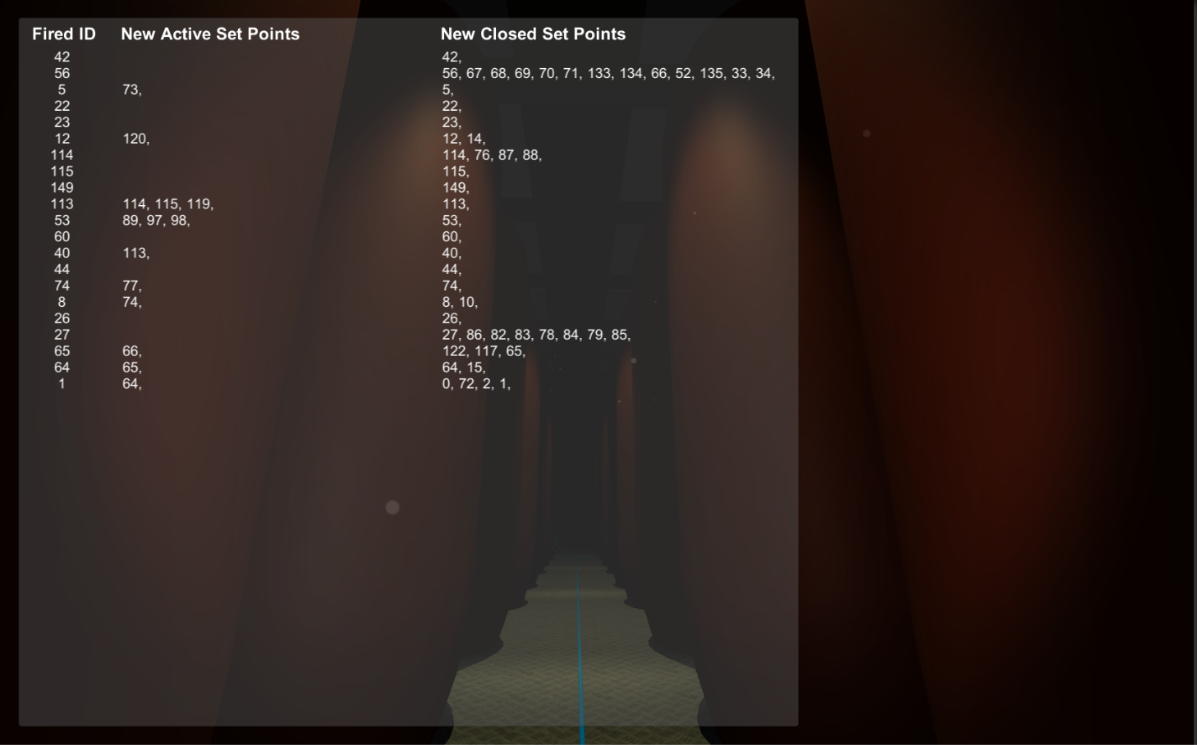
 **Figure 24 – Plot Point 65 is fired, displaying its narrative text on the monitor, followed by information for the tester.**

There are also screen overlays for the tester to observe different changes made within the IA modules such as the graph structure, Player Model, and set structure. Figures 25-29 below display different screen overlays at the same point in a playthrough.



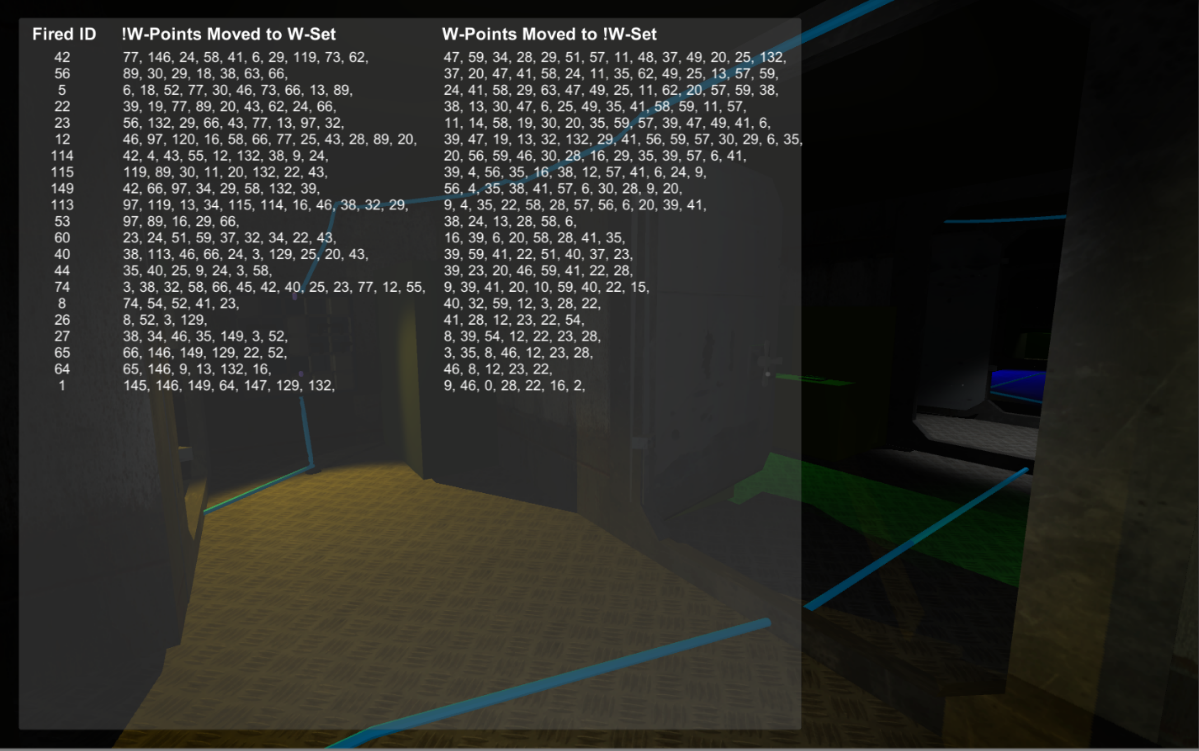
**Figure 25 – Weight change history and similarity of fired weights to player model.**

Figure 25 above shows a screen overlay that contains a history of fired plot point IDs, the points’ weights, the changes made to the Player Model weights and the similarity of the fired weights to the PM. The points are displayed with the most recent fired plot points at the top of the table.



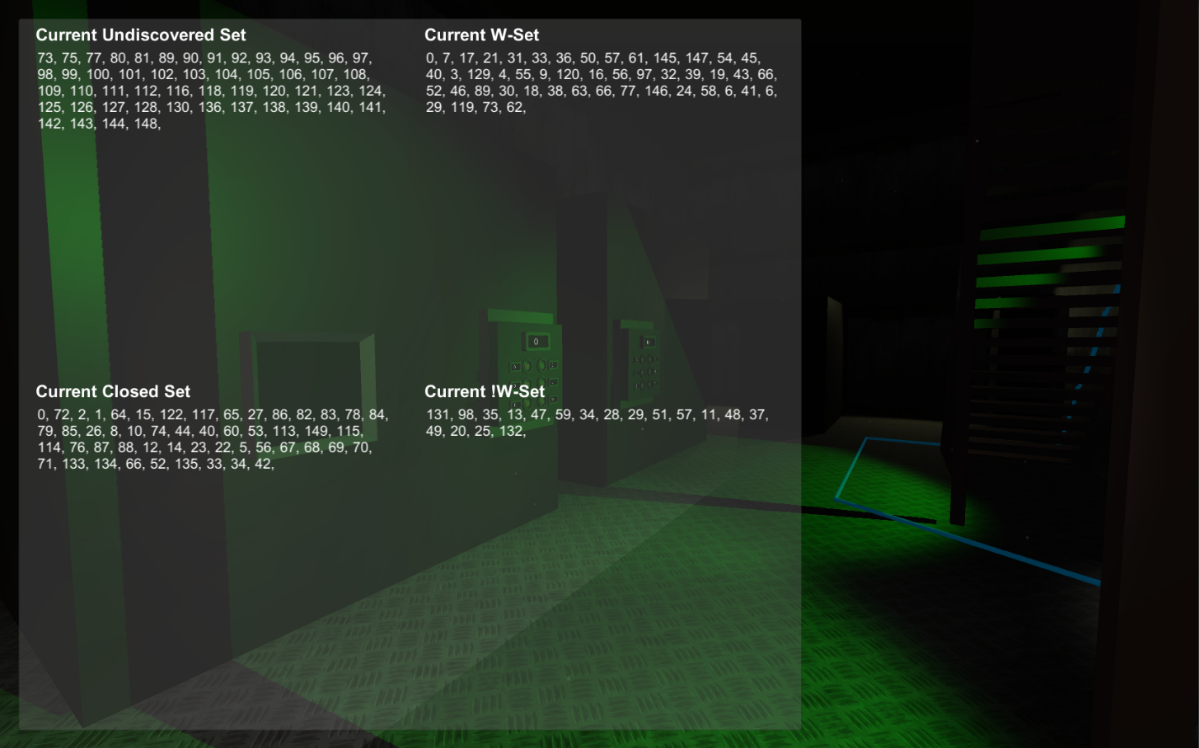
**Figure 26 – Set change history as points in the possibility space are made active or closed.**

Figure 26 above shows how the set structure changes as plot points are fired and the possibility space is navigated.



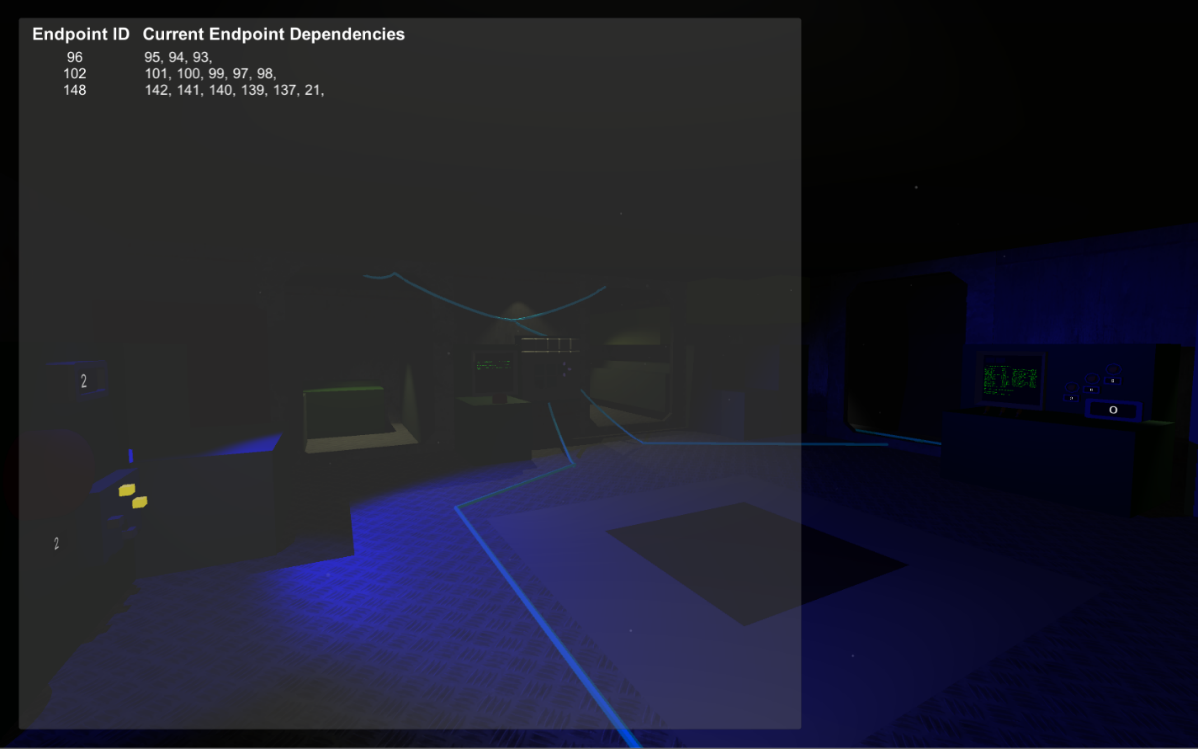
**Figure 27 – Change history within the active set as points move from the W-Set to the !W-Set and vice versa.**

Figure 27 above shows the internal changes within the active set, as plot points move in and out of the world during the replacement process.



**Figure 28 – The current undiscovered, closed, W and !W sets.**

Figure 28 above shows the current contents of each set.



**Figure 29 – Currently non-closed endpoints and their current dependencies.**

Figure 29 above shows the non-closed endpoints and their current dependencies at this point in the playthrough.

## 4.2 Path Traces

The history trace for multiple playthroughs was recorded to text files, which were then used to create tables. The full tables of data can be found in Appendix B.

The playthroughs represented several player playstyles. During some, the tester would follow the plot as directed by the text pages. In others, they would deliberately avoid following any branch, and in others the tester would perform random actions. Through multiple playthroughs, all endpoints were found.

The dataset has 6 endpoints, with IDs 69, 71, 86, 96, 102 and 148. Endpoints 69 and 71 have branches that are very location-specific, with few locations each point can be accessible from. The branch with Endpoint 86 is also location specific but has many paths that lead to the same ending, forming a broad range of possible discourses. Endpoints 96 and 102 are part of a branch that is an extension of that shown in Figure 8. The first points in this branch are only found in one location each, before the rest of the points are potentially found in any of the monitors. Finally, the branch leading to Endpoint 148 is very linear but also begins with a location-specific point, with the remainder able to be found in any monitor.

**Figure 30 – Path A: Shortest Path to Endpoint 69**

Figure 30 above graphs the difference between the fired plot point’s weights to the Player Model weights as the player follows the branch to Endpoint ID 69. The True PM Difference is calculated by subtracting the similarity to PM from the Maximum Similarity to PM to obtain the actual difference. The trendline shows that during this playthrough the true similarity, and thus the preferability, of the points in the branch generally increases with each step. This is because the weight update calculation for the future points in the branch improves the similarity of these later points.

**Figure 31 – Path B: Shortest Path to Endpoint 71**

Figure 31 above shows the shortest path to Endpoint 71. This ending is on the same main branch as Endpoint 69 but is in a different sub-branch. The trend line shows the average True PM Difference is almost constant at approximately 0.04.

**Figure 32 – Difference between Paths A and B**

Figure 32 above shows the only difference in paths is an extra point for path B.

**Figure 33 – Path C: A Longer Path to Endpoint 71**

Figure 33 above shows a longer path to Endpoint 71. The green markers indicate fired points that are part of the branch leading up to Endpoint 71, as seen in the dataset in Appendix A. The red markers show points that are part of branches that lead to different endpoints, and the black markers show points that do not have an endpoint connection. The chart begins with a descending PM difference as the path is followed, before showing a general increase in PM difference over the course of the playthrough.

**Figure 34 – Path D: Endpoint 86**

Figure 34 above shows a playthrough finishing with Endpoint 86. It contains a mixture of near-evenly-distributed red and green points over the course of the path, with one major spike at the 61st plot point fired. In general, the trend line for PM difference stays just above 0.06.

**Figure 35 – Path E: Endpoint 96**

Figure 35 above shows a playthrough that concludes at Endpoint 96. The trendline shows the PM difference increases from an average of about 0.06 to 0.08.

**Figure 36 – Path F: Endpoint 102**

Figure 36 shows the chart for a playthrough that finishes with Endpoint 102. This point is part of a branch that also contains Endpoint 96 but has an effect that forces the path to split so that it continues only to either Endpoint 96 or 102, allowing the system’s endpoint conflict resolution to be evaluated. In this case, 102 was more similar to the Player Model than 96 and thus the branch to 96 was turned off. In this playthrough, there were less spikes in PM difference, and the average difference increased slightly from 0.07 to 0.08.

**Figure 37 – Path G: Endpoint 148**

Figure 37 above shows a path to Endpoint 148. The distribution of red and green points is such that most of the points on the branch were fired in the latter half of the playthrough. The trendline also shows a decrease in average PM difference over the duration.

**Figure 38 – Path H: A Longer Path to Endpoint 148**

Figure 38 above shows a longer path to Endpoint 148 and has more green points distributed in its second half. This shows that as the number of available spaces decreases so does the probability of following this path increase.

**Figure 39 – Path I: No Path Found**

Figure 39 above presents a rare example in which all 64 pages in the environment have been accessed and no endpoint has been reached. This is because of gaps in the dataset and mechanic design, as discussed in the next chapter. The trendline for average PM difference stays constant at approximately 0.06.

**Figure 40 – Average PM Difference as Path Length Increases**

Figure 40 above shows that the average difference between the PM weights and the weights of the fired endpoint tends to increase as the path length of playthroughs increases. This increase appears to be logarithmic.

## 4.3 Performance

The average processing time for the entire DM system process is 66.71ms over 186 recorded actions. This is at an acceptable level for the solution to be considered real-time. The full table of processing times can be found in Appendix C.

# 5 Discussion

This chapter begins with an analysis of the results from the previous chapter, followed by a qualitative discussion on the effectiveness of the developed application.

## 5.1 Quantitative Analysis

Quantitative analysis is a difficult undertaking for this developed application as it becomes difficult to evaluate the subjective nature of the project based on hard figures. However, we can see from the data how the system is operating, and what information informs the IA’s decision-making.

It can be seen from the Dataset preconditions and effects in Appendix A, and the History Tables in Appendix B that the system operates as described in the methodology chapter. The average processing time is also small enough to indicate that the solution meets requirements of operating in real time and indicates that the solution is scalable. However, further optimisation can also be done to the implementation.

The algorithm design and dataset design, and environment design have led to an application that appropriately constrains paths available to the player while allowing different paths to be followed. Thus, the application meets the minimum requirements of the project.

There are, however, a few paths through the narrative environment that lead to no endpoint being discovered. This is due to gaps in the dataset design that result in running out of actions to fire plot points. For example, there may only be 5 remaining pages to read but 6 plot points required to reach a non-closed ending. The system does not account for this. This would be computationally expensive to solve but there are countless design solutions to avoid this scenario. In the case of this prototype, this issue is solved through the implementation of a ‘reusable action’. The current implementation has 64 pages of text that can only be accessed for unique narrative information once before being expended and discarded by the system. A player action that can be revisited guarantees an endpoint is reached, given appropriate possibility space design. This ‘reusable action’ concept also solves several other restrictions of the current system, discussed later in the chapter. In the thematic context of the prototype, this action can take the form of a game mechanic in which the player listens to a handheld radio at different channels. There are other potential approaches that would be appropriate for other games that adopt a similar style of DM system to this project, with the specific strategy dependent on their own requirements.

The difference between the fired plot point weights and the Player Model weights, as charted in the previous chapter, is affected by a very wide range of factors that are not easily tracked. There is a general trend that shows the difference to the Player Model increasing with path length. However, this difference seems to largely depend on the dataset design.

The shortest paths allow the player to follow consecutive points in the branch to the ending, allowing the weight update calculation to improve the similarity of future points in that branch. Thus, the difference to the PM is low. The broader path to Endpoint 86, and the paths leading to Endpoints 96, 102, and 148, have points that are more difficult to find as they could be found in one of several locations and the narrative information does not tend to provide much direction when the range of locations is broad.

In these paths, the player tends to eventually fire the required plot points to progress in the branch. Thus, the greater number of possible locations to find a branch’s plot point in, the more likely it is for the path length of a playthrough to be longer, as it becomes more difficult for the player to find these points. This then influences the difference to PM value, as it becomes unlikely to fire consecutive plot points in the branch, as seen in the roughly-equal distribution of branch points fired in Figure 34, until near the end of the playthrough as the number of remaining possible actions in the world set decreases, as seen in Figures 37 and 38. This has a qualitative impact, discussed later in this chapter.

The logarithmic increase in average difference to PM with path length appears to support this reasoning, as shorter paths are likely those that give good direction to the player, leading to consecutive fired branch points and greater similarity of weights to the PM. It also suggests that large paths with a scattered distribution of points that lead to an ending will have a rather flat trendline, as seen in paths D and I, from Figures 34 and 39, which both have lengths greater than 60. Given this dataset, long paths tend to have a difference to PM of around 0.065. This value is considerably greater than the difference to PM of approximately 0.035 seen in the shortest paths, A and B, seen in Figures 30 and 31.

## 5.2 Qualitative Analysis

Given the subjectivity and complexity of the research topic, the effectiveness of the solution at improving player agency is challenging to effectively gauge with quantitative analysis alone.

During testing, the solution was evaluated by attempting to answer a series of subjective questions:

* Will the player find an ending?
* Is the path to the ending logical?
* How much control does the user feel they have? Does it feel random?
* Does the weight changing algorithm have an impact on the IA’s chosen preferred endpoint?
* Does the PM similarity metric lead to preferred paths?
* How much influence does dataset design have on the output?
* What effect does the number of possible locations for a plot point have on the overall narrative experience?
* How does the size of the dataset affect the narrative?

As discussed previously, there are a few specific paths through the narrative environment that lead to no endpoint being found. This is solved by improved dataset design and by the implementation of one or more reusable actions.

Plotlines appeared more logical in the shorter paths with more direction. The vaguer branches were harder to follow and sometimes gave a greater perception of randomness, but also would occasionally generate a feeling of gratification as the player’s attempts to continue the plot are rewarded.

There were additional qualitative playthroughs in which different longer paths were taken that led to Endpoints 69 and 71, fulfilling all requirements for both. Both endpoints are found in the same location. In some playthroughs the PM was closer to Endpoint 69, and in others it preferred Endpoint 71. The method of annotated weights and a Player Model was able to inform the Drama Manager on how to branch the narrative.

However, there was less perception of agency when taking a longer path, as many points were fired that had no connection to the endpoint reached. This is likely because during a longer playthrough, many points are found that contribute to another separate branch that leads to a very different ending. To an extent, this compromises the narrative integrity of both branches as the story feels more disjointed and less focussed. Great progress may be made through one branch before its narrative information is disregarded in favour of another branch. This is caused by dataset design once again. By trying to branch to very diverse narrative conclusions using soft choices alone, the narrative becomes shallower, and struggles at delivering tone and subtlety.

Dataset design has the greatest influence over the perception of agency in the narrative environment. The system works as intended and is used to effectively traverse a directed graph of plot points. But the effectiveness of the solution is directly attributed to the quality of the data used by the system. This prototype demonstrates how different methods of dataset design influence the perception of agency. Both the size of the dataset and the number of possible locations to access a plot point from, have an impact on the consistency of the delivered story. Potential improvements include a simplification of the possibility space and available actions to allow more focussed storytelling or altering the possibility space data to have sizable variation but less thematically-diverse endings. The dataset should therefore contain more points that are specific in location, and have more subtle influences over the narrative, that eventually add up and allow the DM to branch the story. The graph structure should be clear and not too complicated, with few separate branches leading to different endings as this can take focus away from all branches leading to endings. Therefore, a suggested structure is to have one large yet clear branch that leads to different endings, and few simultaneous smaller plots that branch off from the main directed graph structure, that do not lead to the final ending of the story but allow the Player Model to shift in a designed direction.

To summarize, the developed solution does provide a greater sense of agency that feels earned, but only with branches that have a moderate amount of direction and with limited interference from concurrent narrative plotlines.

## 5.3 Implications

From the results of the project, the usefulness of the project can be evaluated when applied to the commercial games industry. Smaller development studios with fewer resources available for creating a massively diverse narrative environment might choose a similar approach to the project prototype. However, there is a massive amount of work done in setting up test data in the possibility space and the narrative information.

Development time would also be divided between effectively implementing and testing the Intelligent Agent and writing the narrative dataset. Additionally, changes made to plot points can have unexpected results in a large possibility space, further increasing development time. Therefore, the time requirements to debug and tune the DM system and create a robust possibility space can limit the ability to implement this system in a commercial environment. This type of solution should therefore be reduced in scope for smaller projects without the same resources as AAA development studios.

This system can be implemented to great effect within some genres of games. Large open world games such as in GTA V that are sprawling and have great environmental attention to detail, can arguably lack in narrative substance. Many open world games feel somewhat empty, only populated by specific scripted points of interest or things for the player to do but have no further impact. A soft-choice-based Drama Management system can greatly help the appeal of these worlds as a player’s smaller incremental actions can add up to a dynamic narrative that unfolds in such a large environment. This system, and/or variations of it can increase the narrative complexity of a complex game world, making up for any perceived shortcomings. However, based on the quantitative and qualitative analysis, it is suggested that the implementation act as a supplement to more rigid narrative structures implemented by the designer, using a well-balanced mixture of hard and soft choices.

For both larger and smaller projects, the results show that dataset management is a key factor in the effective delivery of player agency.

## 5.4 System Improvements

Aside from improvements to the dataset, the system itself has several areas for improvement.

As mentioned in the previous chapter, a sub-optimal solution was implemented for obtaining the ‘least-viable point’ that would be replaced by the IA as this was a performance bottleneck. This has not had any noticeable effect on the perceived agency of the solution but there exists a trade-off between optimality and speed. However, the annotated weight approach to Player Modelling provides an approximation of the player’s state, and so does not have strict requirements for optimality.

In general, the code can be made more efficient to improve with scalability but is enough for now. The memory requirements in the prototype are sizable, and there are overheads in finding and accessing the information. The possibility size and complexity of trajectory search makes improvements of these attributes desirable for facilitating future expansions to the system. Additional improvements to processing times can be done through methods such as multithreading, but this requires additional care when programming and will likely increase development time for debugging.

As mentioned in the previous chapter, the implementation of ‘reusable actions’ can give the designer a greater ability to create narrative branches with greater direction and an improved structure.

The current system does not consider situations in which two or more points that are dependencies of a shared endpoint only have one available player action between them. In such scenarios, the endpoint is unreachable, but the system is not aware of this until the plot point is accessed through the player action. This has a negative impact on the decision-making process of the DM and could result in wastefulness of the finite player actions. The computational complexity of searching for this type of situation can be very expensive and disproportionate to the frequency that it would occur. Alternative design improvements can be made to avoid this issue, such as improvements to the dataset, and implementing reusable actions.

As discussed in the Literature Review chapter, Drama Management Systems can be combined with emergent IA techniques to provide the ability to adapt and improvise, and DM system approaches such as DDM can add more complexity to NPC characters. Incorporating such techniques on top of the prototype would provide desired behaviour should the narrative environment be developed further.

A great improvement to the system would be the development tools for creating the possibility space, including a user-friendly UI, to enable designers to quickly visualise and for programmers to debug issues. Currently, the possibility space is written in code, making it difficult to find errors and slow to expand. Such tools could also be connected to a relational database that can be queried. The readability of these designer-friendly tools would make dataset management more tenable and can help to eliminate redundancies in the directed graph structure. These tools would take additional time to develop, which should also be a consideration for smaller studios.

The time taken to develop the IA framework, and any extensions or tools, is sizable, which can impact the decision to implement such a system. However, once created, these features can be used for multiple projects and tailored to specific needs. Thus, this can speed up the development of subsequent projects, and allow the system and tools to be improved and iterated on.

# 6 Conclusion

This chapter summarises the most significant reasonings from the previous chapter and presents conclusions before discussing future work.

The developed prototype worked as intended and effectively navigated a possibility space, selecting branches to traverse based on a logical structure and a Player Model heuristic search. In some areas of the directed graph there was a great sense of agency created.

Within the possibility space of the prototype, the plot points with vague location constraints were more difficult to deliberately find, but it is more likely to eventually find its ending as the number of available locations decreases with every action. Plot points with more specific location constraints were more easily found if the player is directed but are also more likely to be discarded after all their possible spaces have been expended.

The most appropriate design solutions for improving the branch structure and ensuring an endpoint is always reached in the prototype application are to implement ‘reusable actions’ in the form of a game mechanic and improving the possibility space structure and narrative information.

With an approach that emphasised soft choices only, dataset design became more difficult to appropriately constrain. A well-structured possibility space is one of the most important factors for the effectiveness of the solution. Branches should be designed with an adequate amount of direction. Branches should not have much overlap or interference from completely separate branches as this leads to a less focused story experience.

Soft choices in a DM system alone are likely insufficient in facilitating a well-structured narrative, and so has limited ability to invoke player agency.

However, overall, this style of Drama Management System offers an excellent supplement to conventional narrative structures, when well-designed and well-constrained. Limitations should be placed on the power of this system over the overall narrative but can still be used to greatly improve the agency of a narrative’s discourse. Therefore, it is suggested that such systems incorporate a mixture of both hard and soft choices to ensure a well-structured possibility space is created. In this format, soft choices would inform the DM on what hard choices to present.

Creating any effective Drama Management system, and any extensions or tools, is a sizable undertaking that requires a great amount of consideration and time in all areas of development. However, once created, it can be modified and tailored to different applications, possibly saving future development time, and allowing the system to be iterated on over the course of several projects.

These solutions are also greatly scalable when the possibility space and narrative design are appropriately constrained, and small concessions are made in favour of speed over optimality.

Player agency in narrative games is a product of many factors. Soft choices in a DM system can improve the perception of agency, when carefully balanced. Designer intent is obscured, and the illusion of choice is improved, but without appropriate constraints, the plot structure can appear less focused.

In conclusion, Drama Management systems can improve the perception of player agency when combining appropriate IA techniques, good game design practices, and a well-crafted narrative structure.

## 6.1 Future Work

The current system can be developed further in accordance with the guidelines described in the previous section; combining hard and soft choices, implementing reusable actions, and improving the possibility space and narrative information.

Storytelling in games is a subjective area of research, with unique goals to each application. This dissertation presents a framework for a soft-choice Drama Management system that can be used as a basis for improvement. Different techniques can be added to this framework to achieve desired behaviour an individual game. DDM can be implemented for games that focus more on NPC behaviour, TTD-MDP can be used where a degree of randomness is desirable, and emergent IA techniques can be used to facilitate improvisation within the DM. Natural language processing could also be used in conjunction with text-based narrative delivery mechanics to perform improvisation or avoid the boundary problem.

Future work in storytelling systems will continue to experiment and combine elements of Drama Management and emergent behaviours, discovering more about the limitations and constraints involved. From this research, best practices emerge and become conventional wisdom.

From this project, approaches regarding soft choices in Drama Management systems have been discussed such that future storytelling applications can apply what has been discovered to improve their effectiveness regarding their individual goals, and these improvements will likely lead to a greater perception of player agency.

# References

Arkane Studios (2012) *Dishonored* [Video Game]. Arkane Studios.

Atari (1972) *Pong* [Video Game]. Atari.

Atari (1980) *Adventure* [Video Game]. Atari.

Atari (1980) *Centipede* [Video Game]. Atari.

Egenfeldt-Nielsen, S., Smith, J.H. and Tosca, S.P. (2012) *Understanding Video Games: The Essential Introduction.*2nd edn. New York, NY, 10001: Routledge.

Gygax, G. and Arneson, D. (1974) *Dungeons & Dragons* [Tabletop Game]. TSR.

Hernandez, S.P., Bulitko, V. and Hilaire, E.S. (2014) 'Emotion-Based Interactive Storytelling with Artificial Intelligence', *AIIDE,*.

Higinbotham, W. and Dvorak, R. (1958) *Tennis for Two* [Video Game]. Brookhaven National Laboratory.

Martens, C. and Cardona-Rivera, R. (2017) *Procedural Narrative Generation.*Available at: <https://www.gdcvault.com/play/1024143/Procedural-Narrative> (Accessed: Nov 11, 2018).

Mateas, M. and Sengers, P. (1999) 'Narrative Intelligence', *AAAI*, . doi: 10.1075/aicr.19.04sen.

Naughty Dog (2013) *The Last of Us* [Video Game]. Naughty Dog.

Nereyek, A. (2000) 'Intelligent Agents for Computer Games, Proceedings of the 2nd International Conference of Computers and Games, Springer, vol. 2063, pp. 414–422.

Nintendo (1981) *Donkey Kong* [Video Game]. Nintendo.

O’Neill, B. and Riedl, M. (2011) 'Toward a Computational Framework of Suspense and Dramatic Arc'*Affective Computing and Intelligent Interaction* Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 246-255.

Procedural Arts (2005) *Façade* [Video Game]. Procedural Arts.

Riedl, M.O. and Bulitko, V. (2012a) 'Interactive Narrative: A Novel Application of Artificial Intelligence for Computer Games', *AAAI,*.

Riedl, M.O. and Bulitko, V. (2012b) 'Interactive Narrative: An Intelligent Systems Approach', *AI Magazine,*, pp. 2013.

Roberts, D., O Riedl, M. and L Isbell, C. (2011) 'Beyond Adversarial: The Case for Game AI as Storytelling', .

Russell, S., Graetz, M., Samson, P. and Witaenem, W. (1962) *Spacewar!* [Video Game]. Russell, S.

Sharma, M., Ontanon, S., Mehta, M. and Ram, A. (2010) 'Drama Management and Player Modeling for Interactive Fiction Games.', *Computational Intelligence,*26, pp. 183-211. doi: 10.1111/j.1467-8640.2010.00355.x.

Steunebrink, B.R., Dastani, M. and Meyer, J.C. (2009) 'The OCC Model Revisited', .

Telltale Games. (2012) *The Walking Dead* [Video Game Series]. Telltale Games

Weallans, A., Louchart, S. and Aylett, R. (2012) *Distributed Drama Management: Beyond Double Appraisal in Emergent Narrative.*Springer, Berlin, Heidelberg, pp. 132.

Weyhrauch, P., and Bates, J. (1997) *Guiding Interactive Drama*. Carnegie Mellon University Pittsburgh, PA.

Wreden, D. and Pugh, W. (2013) *The Stanley Parable* [Video Game]. Galactic Café.

Wreden, D. and Pugh, W. (2014) *The Stanley Parable: A Negotiation of Expressive Choice* [Educational Talk]. GDC. Available from: <https://www.youtube.com/watch?v=lB0OoYX9UfI>

# Appendix A: Dataset

[Dataset Possibility Space.txt](Dataset%20Possibility%20Space.txt)

Available from: <https://drive.google.com/open?id=11B_SjEXajOJizLwQklXq6PWsVR3ZQiL9>

# Appendix B: Full History Tables and Processing Times

<OutputTables.xlsx>

Available from: <https://drive.google.com/open?id=1z32buQaDr06lZJSIvAMC8LcxOsirnnFR>

# Appendix C: Table of Processing Times

|  |  |  |
| --- | --- | --- |
| **Index:** | **Processing Times After Firing Points in seconds** | **Time in ms** |
| 1 | 0.02069092 | 20.69092 |
| 2 | 0.02670288 | 26.70288 |
| 3 | 0.01629639 | 16.29639 |
| 4 | 0.01382446 | 13.82446 |
| 5 | 0.01660156 | 16.60156 |
| 6 | 0.007995605 | 7.995605 |
| 7 | 0.004821777 | 4.821777 |
| 8 | 0.01055908 | 10.55908 |
| 9 | 0.01638794 | 16.38794 |
| 10 | 0.02719116 | 27.19116 |
| 11 | 0.008422852 | 8.422852 |
| 12 | 0.00579834 | 5.79834 |
| 13 | 0.02096558 | 20.96558 |
| 14 | 0.007415771 | 7.415771 |
| 15 | 0.005187988 | 5.187988 |
| 16 | 0.00479126 | 4.79126 |
| 17 | 0.005493164 | 5.493164 |
| 18 | 0.005279541 | 5.279541 |
| 19 | 0.100647 | 100.647 |
| 20 | 0.1054382 | 105.4382 |
| 21 | 0.07348633 | 73.48633 |
| 22 | 0.01663208 | 16.63208 |
| 23 | 0.01182556 | 11.82556 |
| 24 | 0.0105896 | 10.5896 |
| 25 | 0.0113678 | 11.3678 |
| 26 | 0.005874634 | 5.874634 |
| 27 | 0.03405762 | 34.05762 |
| 28 | 0.004318237 | 4.318237 |
| 29 | 0.009399414 | 9.399414 |
| 30 | 0.01643372 | 16.43372 |
| 31 | 0.009689331 | 9.689331 |
| 32 | 0.01266479 | 12.66479 |
| 33 | 0.0118103 | 11.8103 |
| 34 | 0.02018738 | 20.18738 |
| 35 | 0.02832031 | 28.32031 |
| 36 | 0.02957153 | 29.57153 |
| 37 | 0.01493835 | 14.93835 |
| 38 | 0.02107239 | 21.07239 |
| 39 | 0.01443481 | 14.43481 |
| 40 | 0.005966187 | 5.966187 |
| 41 | 0.02383423 | 23.83423 |
| 42 | 0.04208374 | 42.08374 |
| 43 | 0.0128479 | 12.8479 |
| 44 | 0.04629517 | 46.29517 |
| 45 | 0.0100174 | 10.0174 |
| 46 | 0.02364349 | 23.64349 |
| 47 | 0.008144379 | 8.144379 |
| 48 | 0.005962372 | 5.962372 |
| 49 | 0.01699638 | 16.99638 |
| 50 | 0.01561928 | 15.61928 |
| 51 | 0.005615234 | 5.615234 |
| 52 | 0.008178711 | 8.178711 |
| 53 | 0.004882813 | 4.882813 |
| 54 | 0.007263184 | 7.263184 |
| 55 | 0.009216309 | 9.216309 |
| 56 | 0.007263184 | 7.263184 |
| 57 | 0.008666992 | 8.666992 |
| 58 | 0.003356934 | 3.356934 |
| 59 | 0.01263428 | 12.63428 |
| 60 | 0.02935791 | 29.35791 |
| 61 | 0.07891846 | 78.91846 |
| 62 | 0.01971436 | 19.71436 |
| 63 | 0.2763672 | 276.3672 |
| 64 | 0.3239136 | 323.9136 |
| 65 | 0.2460327 | 246.0327 |
| 66 | 0.4851685 | 485.1685 |
| 67 | 0.3746948 | 374.6948 |
| 68 | 0.1705322 | 170.5322 |
| 69 | 0.06018066 | 60.18066 |
| 70 | 0.01037598 | 10.37598 |
| 71 | 0.01135254 | 11.35254 |
| 72 | 0.00769043 | 7.69043 |
| 73 | 0.00994873 | 9.94873 |
| 74 | 0.009979248 | 9.979248 |
| 75 | 0.005584717 | 5.584717 |
| 76 | 0.02032471 | 20.32471 |
| 77 | 0.1399536 | 139.9536 |
| 78 | 0.8050232 | 805.0232 |
| 79 | 0.5157166 | 515.7166 |
| 80 | 0.006225586 | 6.225586 |
| 81 | 0.06676483 | 66.76483 |
| 82 | 0.01126862 | 11.26862 |
| 83 | 0.004211426 | 4.211426 |
| 84 | 0.007568359 | 7.568359 |
| 85 | 0.004699707 | 4.699707 |
| 86 | 0.004943848 | 4.943848 |
| 87 | 0.008850098 | 8.850098 |
| 88 | 0.01196289 | 11.96289 |
| 89 | 0.003479004 | 3.479004 |
| 90 | 0.00390625 | 3.90625 |
| 91 | 0.003479004 | 3.479004 |
| 92 | 0.007843018 | 7.843018 |
| 93 | 0.01480103 | 14.80103 |
| 94 | 0.00479126 | 4.79126 |
| 95 | 0.004821777 | 4.821777 |
| 96 | 0.006561279 | 6.561279 |
| 97 | 0.01174927 | 11.74927 |
| 98 | 0.004821777 | 4.821777 |
| 99 | 0.005554199 | 5.554199 |
| 100 | 0.004241943 | 4.241943 |
| 101 | 0.004058838 | 4.058838 |
| 102 | 0.006317139 | 6.317139 |
| 103 | 0.004379272 | 4.379272 |
| 104 | 0.005889893 | 5.889893 |
| 105 | 0.005813599 | 5.813599 |
| 106 | 0.007629395 | 7.629395 |
| 107 | 0.006347656 | 6.347656 |
| 108 | 0.006118774 | 6.118774 |
| 109 | 0.01637268 | 16.37268 |
| 110 | 0.007339478 | 7.339478 |
| 111 | 0.01446533 | 14.46533 |
| 112 | 0.01164246 | 11.64246 |
| 113 | 0.03056335 | 30.56335 |
| 114 | 0.0135498 | 13.5498 |
| 115 | 0.01013184 | 10.13184 |
| 116 | 1.848793 | 1848.793 |
| 117 | 0.008872986 | 8.872986 |
| 118 | 0.5444565 | 544.4565 |
| 119 | 0.2356834 | 235.6834 |
| 120 | 0.03905106 | 39.05106 |
| 121 | 0.3549538 | 354.9538 |
| 122 | 0.1830902 | 183.0902 |
| 123 | 0.06303787 | 63.03787 |
| 124 | 0.008293152 | 8.293152 |
| 125 | 0.07450199 | 74.50199 |
| 126 | 0.01149178 | 11.49178 |
| 127 | 0.02050781 | 20.50781 |
| 128 | 0.01806641 | 18.06641 |
| 129 | 0.005981445 | 5.981445 |
| 130 | 0.01916504 | 19.16504 |
| 131 | 0.006958008 | 6.958008 |
| 132 | 0.008422852 | 8.422852 |
| 133 | 0.00769043 | 7.69043 |
| 134 | 0.01220703 | 12.20703 |
| 135 | 0.01928711 | 19.28711 |
| 136 | 0.01098633 | 10.98633 |
| 137 | 0.008422852 | 8.422852 |
| 138 | 0.02148438 | 21.48438 |
| 139 | 0.02502441 | 25.02441 |
| 140 | 0.008666992 | 8.666992 |
| 141 | 0.0177002 | 17.7002 |
| 142 | 0.009887695 | 9.887695 |
| 143 | 0.01446533 | 14.46533 |
| 144 | 0.04107666 | 41.07666 |
| 145 | 0.02276611 | 22.76611 |
| 146 | 0.01184082 | 11.84082 |
| 147 | 0.008056641 | 8.056641 |
| 148 | 0.006164551 | 6.164551 |
| 149 | 0.008666992 | 8.666992 |
| 150 | 0.006225586 | 6.225586 |
| 151 | 0.007080078 | 7.080078 |
| 152 | 0.007263184 | 7.263184 |
| 153 | 0.007080078 | 7.080078 |
| 154 | 0.01116943 | 11.16943 |
| 155 | 0.008972168 | 8.972168 |
| 156 | 0.02087402 | 20.87402 |
| 157 | 0.03857422 | 38.57422 |
| 158 | 0.01043701 | 10.43701 |
| 159 | 0.02697754 | 26.97754 |
| 160 | 0.03045654 | 30.45654 |
| 161 | 0.01519775 | 15.19775 |
| 162 | 0.01446533 | 14.46533 |
| 163 | 0.01513672 | 15.13672 |
| 164 | 0.01550293 | 15.50293 |
| 165 | 0.03118896 | 31.18896 |
| 166 | 2.170334 | 2170.334 |
| 167 | 0.03746033 | 37.46033 |
| 168 | 0.03230286 | 32.30286 |
| 169 | 0.02674866 | 26.74866 |
| 170 | 0.03569031 | 35.69031 |
| 171 | 0.03422546 | 34.22546 |
| 172 | 0.2748413 | 274.8413 |
| 173 | 0.2161407 | 216.1407 |
| 174 | 0.01713562 | 17.13562 |
| 175 | 0.1961746 | 196.1746 |
| 176 | 0.1786041 | 178.6041 |
| 177 | 0.01755524 | 17.55524 |
| 178 | 0.01560974 | 15.60974 |
| 179 | 0.01034546 | 10.34546 |
| 180 | 0.03450775 | 34.50775 |
| 181 | 0.006004333 | 6.004333 |
| 182 | 0.004203796 | 4.203796 |
| 183 | 0.009441376 | 9.441376 |
| 184 | 0.006631851 | 6.631851 |
| 185 | 0.08029842 | 80.29842 |
| 186 | 0.01135826 | 11.35826 |
|  |  |  |
| Average Time: | 0.06670756 | 66.70756018 |