Emotional Affect from Procedural Content Generation

By

Eóin Fintan Joseph Murphy

Supervisor: Mads Haahr

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfilment of

the requirements

for the Degree of

Master in Computer Science

University of Dublin, Trinity College,

May, 2017

Declaration of Authorship

I, Eóin Murphy declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not been previously submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signed: ______________________________

Date: ______________________________

Trinity College Dublin

Abstract

Integrated Computer Science

Computer Science and Statistics

Masters of Computer Science

Emotional Affect from Procedural Content Generation

Procedural Content Generation (PCG) is a very successful tool that has been widely used in the videogame industry. While the tools have been used for various purposes such as creating dynamic textures, characters and even music it is primarily limited to creating levels and environments within games. This does create unique and interesting worlds but does not necessarily engage the audience.

Previous work in this area has attempted to create puzzles and challenges for the players using these methods. This dissertation tries to go one step further, to guide the content generation using an emotional model to capture the emotional state of the player and how it changes. By targeting the emotions of the player and altering them through gameplay; richer and more engaging content can be created.

In this dissertation Russell's two-dimensional model for emotions is combined with a puzzle generating engine to guide the content created for the player, based on their emotions. The result is a search through the game mechanics for gameplay that both achieves some local goal and engages the player on an emotional level.

This opens up the field for further investigation while giving developers more creative control over these procedural content creation systems.

Summary

This dissertation concerns the work of procedural content generation (PCG) in videogames and augmenting PCG with emotional models. It recognises the importance and the potential of this technology but looks at its limited application. It establishes the varied use of PCG in industry and the potential pitfalls of its use. Particularly, the problem of creating environments in PCG without the narrative or gameplay elements to engage the player.

A brief background is given on the use of PCG to create puzzles, specifically narrative puzzles that can engage the audience with PCG content. Combined with this is a background on narrative and emotions and their use and importance in video games. Emotional models are given due consideration for their ability to model players and affect player emotions. Affective computing is noted as being a rare subject of research but with great potential.

Russell's model of emotions is examined and used for emotional modelling because of its expressive power. The axes of Russell's model: valence and activity are intuitive for relating emotions to the emotional space described and therefore are used for modelling player affect.

A new puzzle creation system is designed by combining Russell's emotional model with De Kegel's narrative puzzle generation system. This new system allows for PCG to create scenes within a game that can affect the player emotionally by implementing known rules and affects. These can be combined in sequence to create a larger emotional narrative.

The system implemented is described, explaining how Unity and C# were used to incorporate the emotional model into an existing puzzle generation system.

The system created is analysed with its limitations, particularly its limited scope and narrative depth, discussed. Overall the area of PCG, particularly for narrative and emotional content is found to be an open area with little discussion present. The possibilities for future work are discussed with note given to the need for tested research in this area.

Acknowledgements

I would firstly like to thank my friends and my family, who have supported me through thick and thin. Without you I would not have made it to this stage. So, for all your help and encouragement, thank you. I would also like to thank my supervisor Mads Haahr for his help and advice which helped shape this dissertation.

Table of Contents

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Objectives 3

1.3 Roadmap 4

Chapter 2 Background and Related Work 5

2.1 Procedural Content Generation 5

2.1.1 Procedural Content Generation for Narrative 6

2.3.1 The Narrative Puzzle System 8

2.2 Narrative Structure 10

2.3 Emotional Affect and Emotional Models 14

2.4 Emotions and Narrative in Games 19

Chapter 3 Design 21

3.1 The Emotional Model 21

3.2 The Augmented Grammar 24

3.3 The Puzzle Creation System 26

3.4 The Search System 27

Chapter 4 Implementation 28

4.1 The Game World 28

4.2 The Puzzle Generator 29

4.3 The Unity Editor 31

Chapter 5 Evaluation 33

5.1 Game Demo 33

5.2 Using the Model 35

5.3 The Emotional Model 36

5.4 Comparison with Previous System 36

Chapter 6 38

Conclusion 38

6.1 Main Contributions 38

6.2 Limitations 38

6.3 Future Work 39

6.4 Final Thoughts 41

Bibliography 42

Chapter 1

Introduction

This dissertation explores the role of Procedural Content Generation in the modern videogame industry. Procedural Content Generation (PCG) is a popular tool employed in video game development and is a technique that has been used since the earliest days of the industry (*Rogue (1980)*, *Elite (1984))* [1] . PCG is the automated creation of content through some predefined process. The developers create their game engine and define a method to (often randomly) create game elements such as enemies, terrain, obstacles etc. This is often done by creating a randomly generated seed which is then fed into the content creation process. This enables game developers to create far more content than they could possibly enumerate by themselves. Instead of finishing a game with its entire content already defined; the game will create textures or levels when it is running.

This tool is vital for certain games such as strategy games where the game mechanics are static and essentially gameplay is the same scenario. The key to its success is the ability to tweak parameters and the random generation; which together create interesting gameplay and subtle nuances that separate one experience from the next. Similarly puzzle games can make extensive use of PCG to create a near infinite number of levels for their audience.

1.1 Motivation

The primary motivation for this research was to find some form of content creation that would be more directed than current systems. Looking at the current industry examples PCG is primarily used for creating random items for the player with randomly generated statistics and properties such as in *Diablo III (2012)*. Alternatively, it has been hugely successful with creating game worlds and environments. For example, *Dwarf Fortress* [2], a two-dimensional game with randomly generated worlds. Its spiritual successor *Minecraft* [3], which similarly creates entire worlds with varying biomes. Using PCG it creates diverse landscapes such as tundra, mountains, deserts, and dense jungles.

Most recently *No Man’s Sky* [4] made heavy use of PCG to create an entire galaxy of stars and planes and even ecosystems on planets. While this was incredibly impressive and garnered much publicity for the game, it left its players with much to be desired. Once the novelty of a new environment had worn off; the players realised that the underlying game was void of content. This led to wide criticism of the game and although it was not a commercial failure it led to heavy criticism for a game that was heavily anticipated to be a success.

Figure 1: No Man's Sky procedurally generated beautiful worlds, but its lack of gameplay upset players

There is a stark distinction between games which are highly praised for compelling narratives, and those which create large open worlds. They achieve this mostly through the use of PCG (in some form). Compelling narratives are carefully crafted, often with linear narratives and with large oversight by the game developers. One huge advantage of PCG is its ability to create unique content that increases replay value and leads to unique experiences across playthroughs.

Games with compelling narratives have been around for a long time. These are usually created manually and meticulously by the developers. This is a slow and careful process that prohibits the use of PCG to ensure that the player has the desired experience. The result is akin to a skilled director in filmmaking. They know what the audience will see and can gauge their reaction. This is used to great effect to play on the emotions of the audience. There is an inability to create content that is both dynamic and emotionally powerful.

The motivation here is to create content that engages the player, that provides some control over the narrative the player experiences. Most importantly to do this without sacrificing the ability of PCG to create unique content. The purpose of this dissertation is to address the issue that is common within industry, that of content with no real substance. It is insufficient to create content for the audience; the audience must be engaged with the game.

By utilising the player's emotions games can create more interesting and enthralling stories which adds both to the artistic value and the commercial viability of a game. While it is not easy there are techniques that can be used to (reliably) alter a player's emotional state and thus keep them invested in the world. Unfortunately, there is a lack of research into narratives in games and emotions or emotional modelling in games. This led to the conclusion that this is an unexplored research topic which could have great potential.

1.2 Objectives

The central aim of this dissertation is to create a system that can combine the content creation power of PCG with a system that guides the generation process to create more interesting experiences for the player. The goal in this research is to create interesting narrative scenarios that the player can engage with on an emotional level, but to create them in a procedural fashion.

The objective is to create a system for level creation that allows the game developer/designer to state how they want the player to feel at certain stages of the level. The content can be dynamically generated to suit this restriction. When this is combined in sequence the system will give the developer the ability to determine the flow of the narrative and alter it as they see fit. The system should not be cumbersome or overly complex as it will need to be used without a detailed understanding of the background mechanics that underlie the system.

An additional goal of this work is to explore new uses for PCG and to further discussion in this area. This dissertation aims to build on the current work in the field of PCG and emotional modelling and to encourage or even inspire future work to develop this field.

1.3 Roadmap

This dissertation will begin with the background to this work in Chapter 2. The background constitutes two different areas of research, that of procedural content generation and that of emotional affect. Chapter 3 will move onto the design for the procedural generation engine and how it combines these areas of research. Chapter 4 will discuss the implementation details of the research. In chapter 5 the results of the work will be explored and finally chapter 6 will discuss some conclusions from this work and explore the potential for future work in this area.

Chapter 2

Background and Related Work

In this chapter the background work for this dissertation will be laid out and explained. The work here will lay the foundation for the rest of this dissertation and will provide further context to this area. There is substantial background material because there was significant work in researching this topic and because it combines two mostly separate fields of work. There is the background section on PCG and its use in games and for narrative purposes. Following that is the background for emotional models and how emotions can be captured in a succinct manner. This will then blend into the background work on emotions within games and the research into this area (or lack thereof).

2.1 Procedural Content Generation

PCG has been used to great commercial success in numerous industry titles. It is a reliable method to create content continuously preventing players from growing bored with encountering the same scenarios repeatedly [5] [6]. One type of games known as 'roguelike' games (due to their similarity to the original *Rogue*) are very difficult and losing the game results in an entirely new game world being created. It is difficult to master these games because of their difficulty and the randomness added by PCG creating the levels anew each time. This of course is the appeal, because they can provide hours of entertainment from simple engines combined with PCG.

Procedural content generation usually creates content in one of two methods: generate-and-test or constructive algorithms. As the name suggests, generate-and-test methods create content randomly and then apply a series of checks to ensure that the content satisfies the criteria given. For example: a puzzle creating engine for a game like Sudoku could create a random grid of numbers and then check for the validity of this grid. The alternative method, constructive algorithms builds the content up in a series of stages such as using a Markov chain. At each stage validity checks are performed to ensure that the content is satisfactory.

PCG creates random or pseudo random content. Thus, the developer does not specify the exact experience the player has, instead the developer creates the rules, the criteria for the content and lets the algorithm create the content from this. These criteria can be anything from difficulty (more abstract) to the number of rooms in a building (more concrete). It can get more complex as the number of parameters grow, for example in creating landscapes and requiring certain types of terrain, sufficient water, etc.

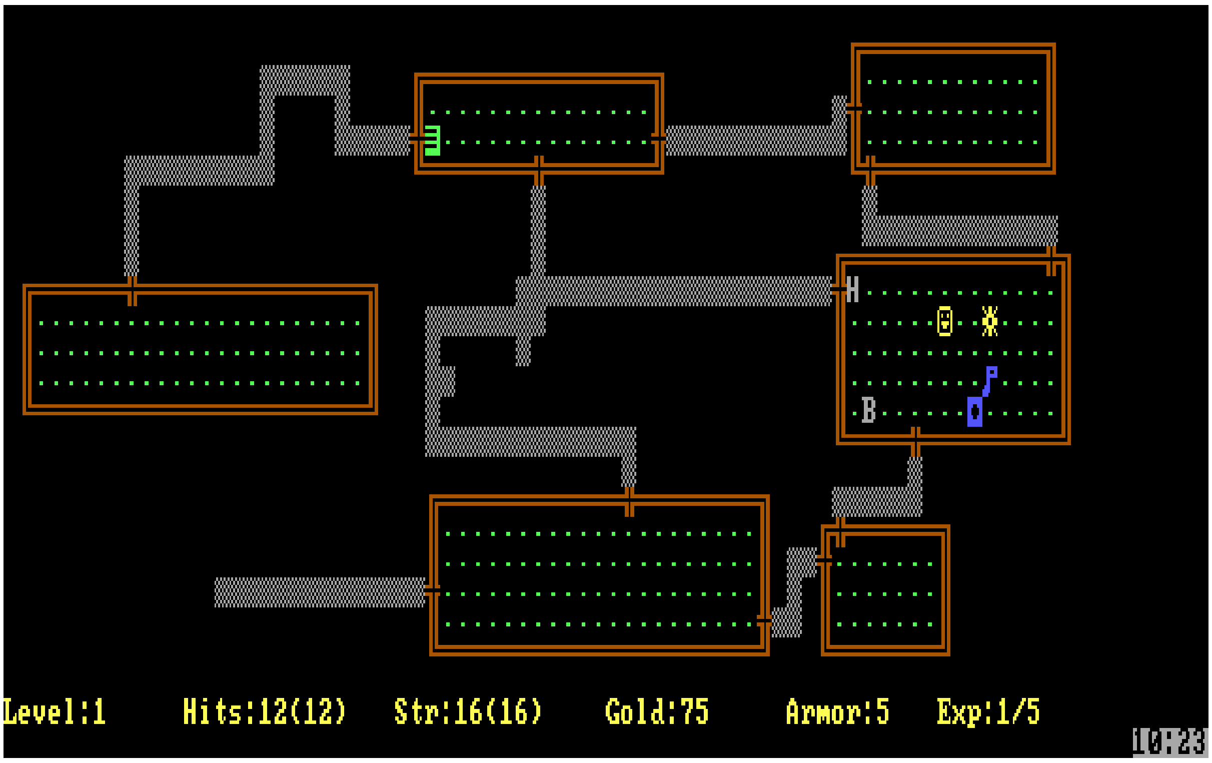


Figure 2: Rogue (1980) one of the pioneers of PCG, each game created a new world ensuring new content continuously

2.1.1 Procedural Content Generation for Narrative

PCG has been used to a limited extent to create narratives for the player, or more accurately add narrative quests to a game dynamically. In *Skyrim* [7] for example a system was used to create so called ‘radiant quests’. These were randomly generated quests that non-player characters (NPCs), the computer controlled agents, would give to the character. However, they were very limited in variety, they were mostly ‘fetch quests’ that would ask the player to go to point A, collect an item, and then go to point B. Additionally the rewards could be randomly generated. While this was made more interesting by sending the player to locations they had not visited; it was repetitive. In the case of *Skyrim* the PCG use was very limited relying on most of the narrative to come from hand crafted storylines which remained diverse and intricate.

The challenge is to develop a story algorithmically, which requires some understanding of narrative structure. The main work in this area was considering procedural generation of narrative puzzles. De Kegel [8] worked on attempting to create puzzles that would fit into an overarching narrative or storyline. While much of that work was focused on the puzzle generation it was important step to finding a system that made content with a purpose. The narrative puzzle was itself heavily influenced by the Puzzle-Dice system [9] which decomposed goals and objects into prerequisite goals and items. This makes intuitive sense, before you can unlock a door you must possess a key. This door may be preventing the player from progressing to the next level/scene in a larger narrative. While the game designer can choose to have this goal, the method of achieving it can be generated procedurally. For example: the player could be tasked with breaking the door, picking the lock or finding the key.

This can contrast or improve upon the generic fetch quest. Thanks to PCG the tasks the player must complete could vary wildly and ensure new combinations and experiences. While they might still have to retrieve an item X from point A they may have several sub goals to complete to get to that stage with potentially different solutions.

2.3.1 The Narrative Puzzle System

The system used represents puzzles using a ‘puzzle tree’ this is very simply a tree showing the decomposition of a goal into the items and stages required to achieve it. The puzzle tree determines the correct sequence of actions required by the player in order to complete the puzzle. The decomposition is determined by many rules which essentially describe the game mechanics. Rules determine what is and is not a valid action to take. Rules operate on the items in the game and describe transitions from one state to the next (how a door goes from locked, to unlocked). A sample puzzle tree is shown below in figure 3. This shows the possible scenario where a player must get outside, getting past a locked door. Here the actions taken to progress area annotated at each stage, with the player moving from right to left. The primary goal for the player is to open the door, to do this they must climb a ladder and get the key. It can be seen how the items can be decomposed to create a more detailed puzzle. It is not difficult to imagine how the items can be decomposed further e.g. how the player obtains the money or the ladder could be elaborated on.

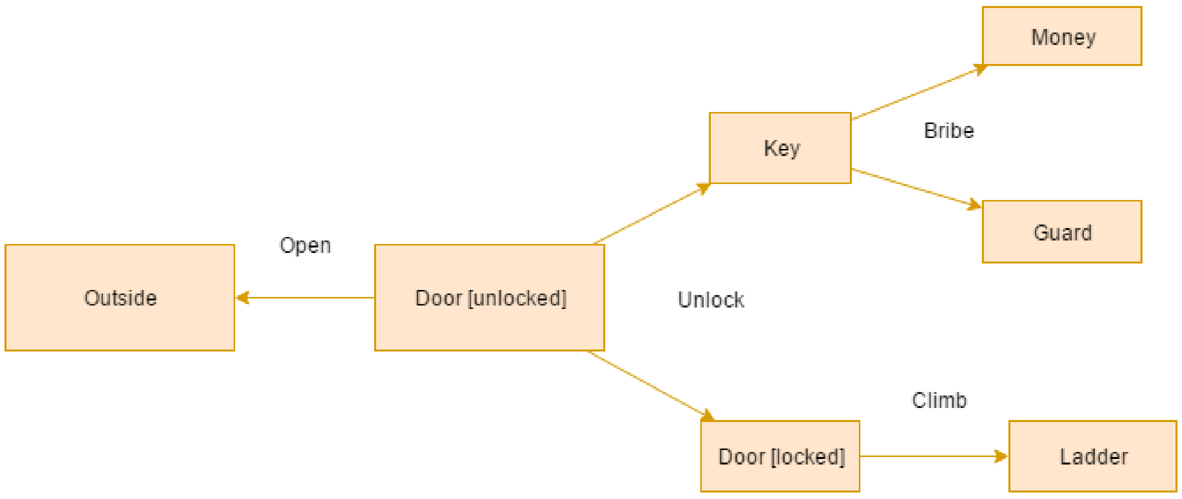


Figure 3 : Sample puzzle tree to get out of a locked room

Each rule has several input terms, an action term and several output terms, one of which is the main output term. This is depicted in figure 4 below. The terms themselves may have several properties determining what game item they refer to and other metadata about those items.

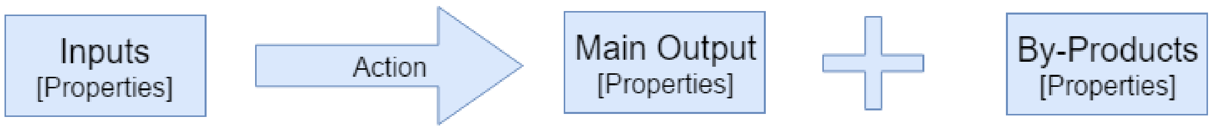


Figure 4: Format of a rule

The rules form a type of context free grammar where the terminals are the final items that the world is populated with and the rules are applied to the non-terminal elements until there is a satisfactory depth to the puzzle. The puzzle generator starts with the goal and recursively applies rules to the items to generate the puzzle tree and thus the puzzle. There are six key elements to this system:

- Game Areas: The game environment is divided into these areas which segregate puzzles. Each game area has a goal that must be achieved before progress is enabled.
- Rules: These are the principle component of the puzzle generation system and dictate what actions are allowed. Rules consist of a list of input terms, an action and a list of output terms.
- Terms: The basic unit that composes rules. Each term has a type e.g. 'tree' and a list of properties (optional)
- Properties: Simple data about terms, they consist of a name, a type and a value e.g. age: (int) 25.
- Action: The action the player takes to apply a rule
- Items: The in-game items, they have a set of properties and refer to objects in the game world.

The puzzle generator maintains a database of in game items and rules that are consulted in the puzzle generation process. These must be supplied by the game developers and provide the building blocks for the entire game/puzzle system.

This system is flexible enough to allow developers to edit and add to the existing system to create various narrative puzzles. The system is independent of any specific implementation and once in place it can be reused for various games. By adding rules and items to the existing pool the players can be provided with new experiences with minimal effort on the developer's part. This method is constructive; it ensures that the solution is allowed by the game rules and that the player has the necessary tools to reach the goal (populating the world with any missing items required). Unfortunately, it does not check that there remains a viable solution. It is possible for a player to take incorrect actions that render the puzzle unsolvable and thus fail the puzzle. One limitation with this system is that it does not track such events.

2.2 Narrative Structure

A narrative, or simply a story, is a collection of connected events combined together into a single plot. Narratives tend to (but do not always) follow a simple structure. Everyone is familiar with the idea of having a beginning, middle and end. This three-act structure is used in most media and has come to be expected. It is the de facto standard for most stories and indeed for writing in general. This structure is similar to Freytag's pyramid (shown in figure 5), [10] which follows a five-act structure but has a similar layout. Freytag's model shows the tension or action in a story over time. It goes from the beginning of the story, with some inciting incident which starts the plot. This leads to increasing action throughout the story until a climax is reached. Then the activity falls as the resolution of the story is reached, then the story enters the denouement where the plot is wrapped up and the story concluded. Other narrative structures exist such as Syd Field's paradigm (the more traditional three-act structure) etc.

One interesting technique is 'in media res', this is the pattern which starts a story in the middle of the action and tension and then goes back to the inciting incident, explaining how the story unfolded to the current events. Whilst various methods can be used there are established patterns and structures that are rarely deviated from or when they are often create new patterns that can be just as recognisable.

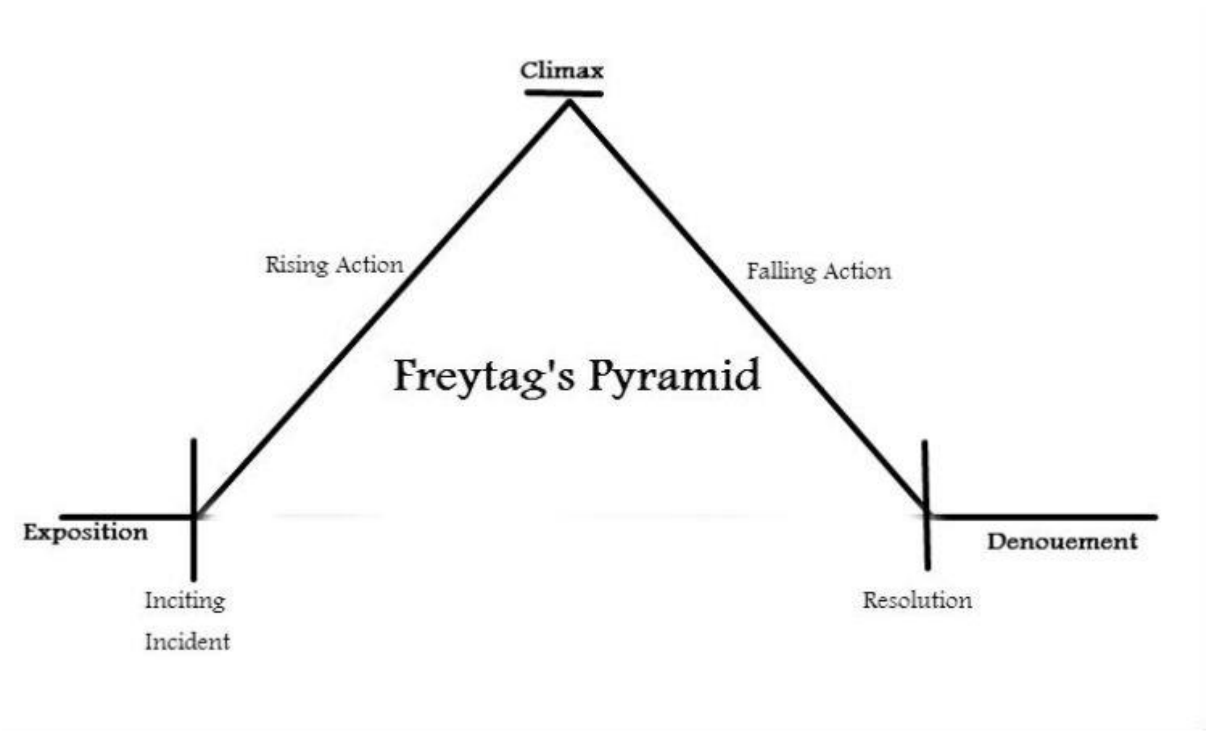


Figure 5: Freytag's Pyramid

A similar but more complex pattern is exhibited in the Star Wars film [11]. Again, the tension of the story is plotted over time, however in this case the subtle nuances of individual scenes has been further explored. This makes it clear how exactly the story flows and how periods of relief are given in order to allow the audience to relax somewhat while the overall tension grows. This is depicted in figure 6. The graph shows how the film starts with an immediately captivating scene with high action adding tension and grabbing the attention of the audience. This is immediately followed by a lull that allows the audience to settle down and immerse themselves in the film. At several key moments, the story advances with significant events increasing the tension: the death of Luke’s family, the use of the Death Star to destroy a planet, the death of Luke’s mentor Obi-Wan and of course the climactic fight at the end of the story. The film is clever enough to give breaks between these important events and not to just increase the excitement constantly. This makes the scenes more effective and gives them added weight. This structure is more detailed than Freytag’s pyramid but exhibits the same properties. It can be applied to many stories in various media, Star Wars is just the example given.

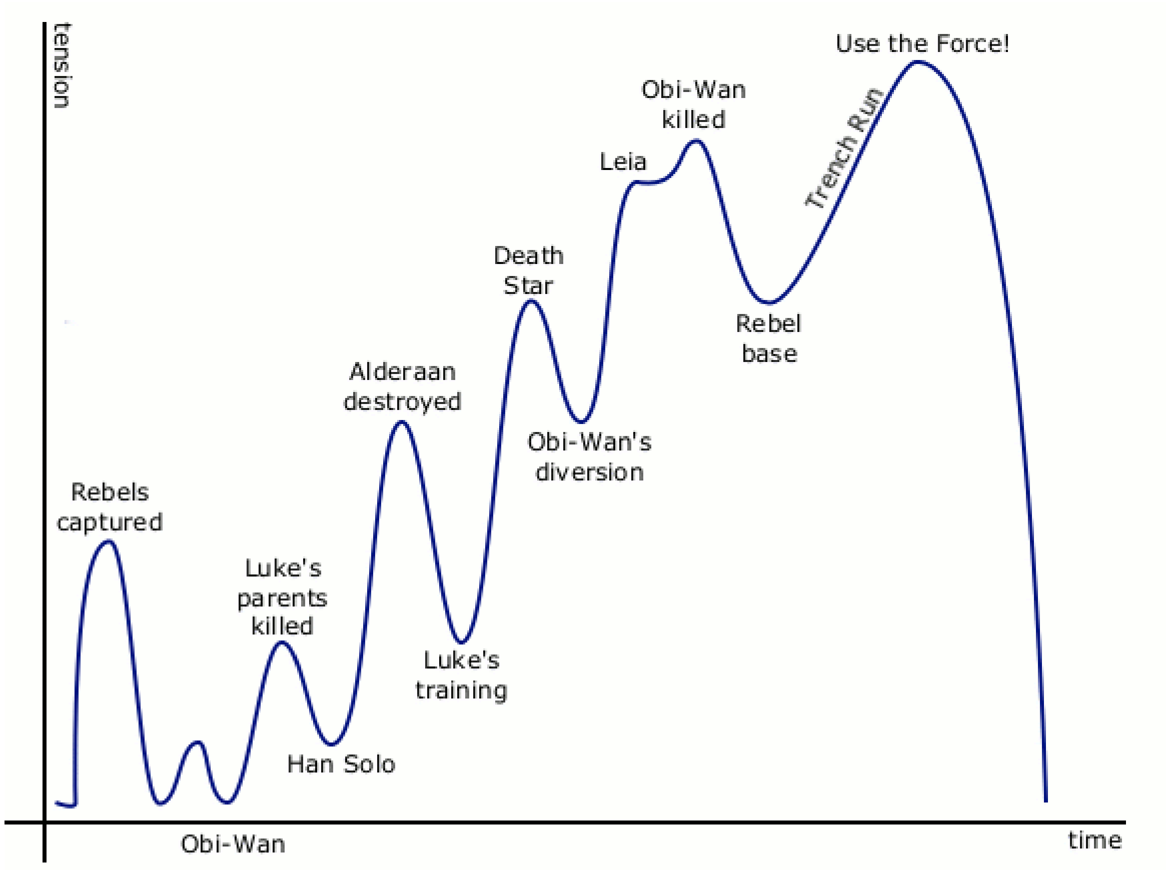


Figure 6: Tension in the storyline in the film Star Wars

There are examples of games that use narrative engines to create believable stories that can vary between iterations of the game. *Façade* [12] used a narrative engine and speech synthesizer allowing the player to use natural language when interacting with the characters. It contained several story ‘beats’ or key points that would trigger at various stages. The aim was to balance tense moments with intermittent breaks while steadily increasing the tension in a similar manner to Freytag’s pyramid.

Narrative engines have been used successfully the AI ‘director' in *Left 4 Dead* [13] tracks the intensity of action for each of the player characters. When they are ‘relaxed’ the director adds more tension and conflict by spawning more enemies for the players to fight. This behaviour is shown in figure 7 in the middle graph. When the players have been in tense situations they are given time to relax with the director easing the number of enemies and giving the players items. This gives the game some pacing and allows the players some recovery time before the tension rises again. This has proved very effective and thanks to PCG leads to diverse gameplay despite the actual layout of the levels remaining the same. This has greatly added to the replay value of the game because players get a different experience each time but with reliable narrative behaviour. While *Left 4 Dead* only acts on one axis (tension) it is not hard to imagine how this method could be expanded for more complex narrative engines.

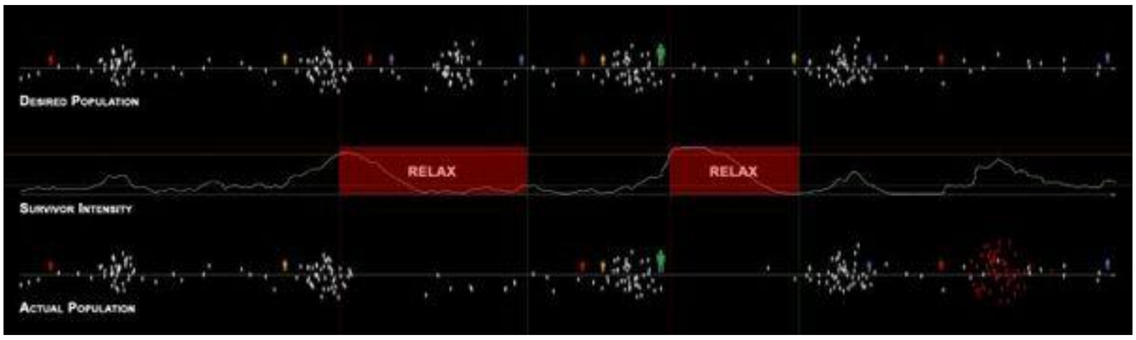


Figure 7: Diagram of director's behaviour in Left 4 Dead over time, tension is shown in the middle

It can be seen from established methods of storytelling that narratives can be examined and their structures can be accurately portrayed and reused between stories. Some game engines have tried to utilise this predictability in order to make their narratives more effective. *Left 4 Dead* uses PCG and narrative structure to great effect, controlling pacing for the enjoyment of the player. The AI director knows when to add a pause and when to ramp up the action, but can vary its behaviour, sometimes starting with high action (much like ‘in media res’) and other times leaving the action to a climactic final scene. The use of PCG keeps players on their toes and adds extra excitement which is harder to capture in static games where the encounters can be memorised and lose their effect over time.

The significance in the narratives described is that they are necessarily structured. Where there is structure there can be algorithms to create that structure. There is evidence that PCG can and has been used to create narratives. If this is controlled by the developer, then they can use PCG to create interesting stories whilst maintaining control over the player’s experience and maintaining the benefits of PCG.

2.3 Emotional Affect and Emotional Models

Affect is a term used in psychology and it will be used repeatedly in this dissertation. Affect, simply put, is emotion. It is defined as the experience and feeling of emotions. Emotions themselves are a nebulous concept, although there is a common understanding of what an emotion is it is hard to define. “Everyone knows what an emotion is, until asked to give a definition. Then, it seems, no one knows” [14] Hunger for instance is not an emotion but it can elicit emotions such an unhappiness, anger etc. Yet the feeling of hunger is as tangible as feeling joy or sadness. Nonetheless there have been many attempts to understand emotions within humans. Some have proposed that emotions are the interpretation of physical stimuli e.g. shortness of breath, fast heart rate, etc. are understood as anxiety. However, it has been shown that physical reactions can be induced in the body, and although the subjects feel the effects they report no emotional response. Although emotions are clearly linked to physical reactions, and are always accompanied by them the reverse is not true. A person can have a physical response without an emotional one. It is assumed that the reader understands what emotions are without having to state it explicitly, which is a difficult task. For the purposes of this work it is not necessary to know exactly what constitutes an emotion but rather how to elicit them within others i.e. how to create affect, and how to model this behaviour.

For the purposes of this work it is necessary to model the emotions of the player as they play the game. Unsurprisingly there have been many models proposed for emotions in humans. There are the simple representations of emotions on the scale of 1 to 10 or using a Likert scale. One theory that is commonly espoused is that of several ‘base’ emotions from which all others are derived through combinations of varying intensity. This concept can be seen in Plutchik’s wheel of emotion (depicted in figure

8). In this view there are a small number of basic emotions that can be combined. This is a somewhat simplistic representation that hides the complexity of emotions and is also unable to fully capture the spectrum of emotions that can be experienced. Althought Plutchik’s wheel is a pleasant diagram of emotions it is hard to quantify in concrete terms.

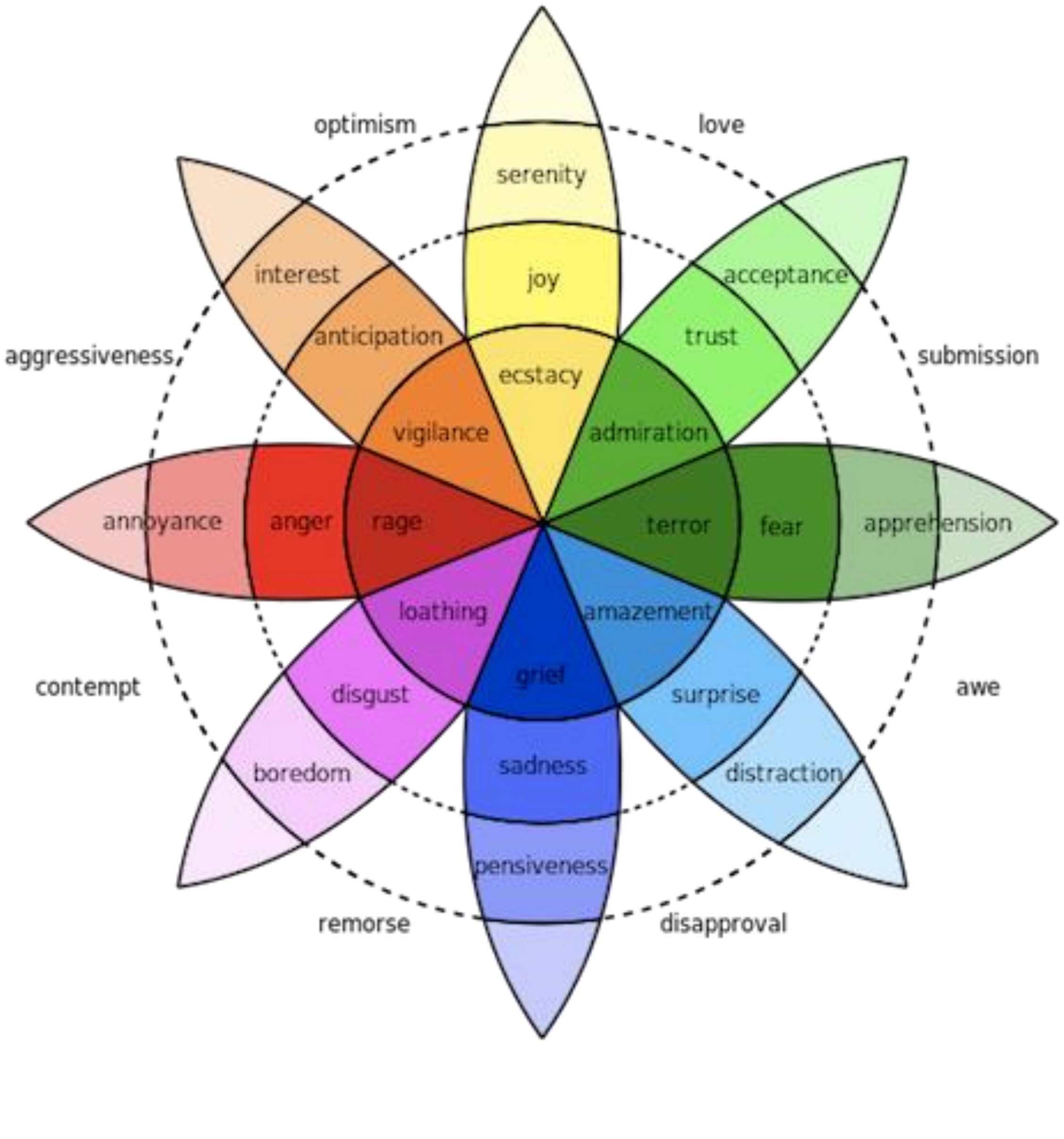


Figure 8: Plutchik's Wheel of emotions

Millenson looked at emotions that were observed in other animals, he drew heavily from Watson's earlier studies. Watson believed that the primary emotions were unconditioned and that the basic three emotions were fear, rage, and love [15]. Millenson agreed with this theory and created a three-dimensional model with the principle axes as: ecstasy, terror and rage (depicted in figure 9). Along these axes there are varying levels of each emotion e.g. pleasure, elation and ecstasy are different intensities of the same emotion [16]. This model (he argued) can represent emotions as points in this three-dimensional space.

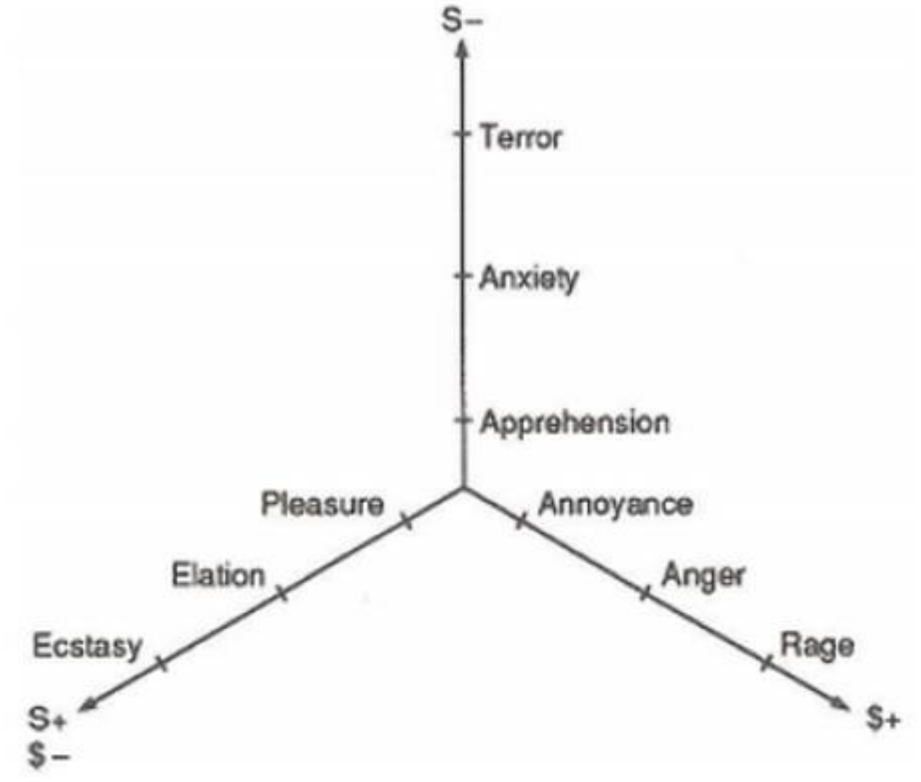


Figure 9: Millenson's Model

Millenson's model was also the basis for an emotional model using fuzzy logic to measure emotional states [17]. Points in the system (based on Millenson's model) represent emotional experiences, and using fuzzy logic each point has a probability of representing an emotion e.g. probabilities for being anxiety, sadness and anger are: 1.0, 0.3, and 0.0 respectively.

This model has some difficulties; most importantly is that it has limited use. While it excels at modelling the three emotions on its axes it is not good for modelling other emotions. Trying to determine where an emotion such as boredom or impatience lie in this space is very difficult and could be debated at length. The second and most significant difficulty is that it is hard to understand vectors in this space. What feeling would be expressed by (+1, +1, +1)? Millenson's model works well for more qualitative work but for quantifying emotions, a challenging endeavour as it is, it is not very helpful.

An alternative approach was taken by Russell who abstracted from emotions two principle axes [18]. Russell uses the axes of valence (how positive or negative an emotion is) and arousal (or interest). His circumplex model is shown in figure 10. It can be seen how negative activation corresponds to low energy feelings such as fatigue or relaxation (showing such emotions can still be positive). While the valence of an emotion determines its pleasantness or how good or bad an emotion is. One of the great strengths of this model is that activation is very similar to tension which was important when examining narratives and narrative structure which presents a good overlap between the emotional model and the narrative structure examined in games. Russell's model is easier to conceptualise because it allows a broad range of emotions, furthermore the diagram instantiates the positions of several key emotions. Not only does this provide those emotions with established positions but it also gives meaning to the various vectors in this model. A change of (-0.5, 0.5) can be seen as calming, moving the emotion closer to serenity and away from frustration.

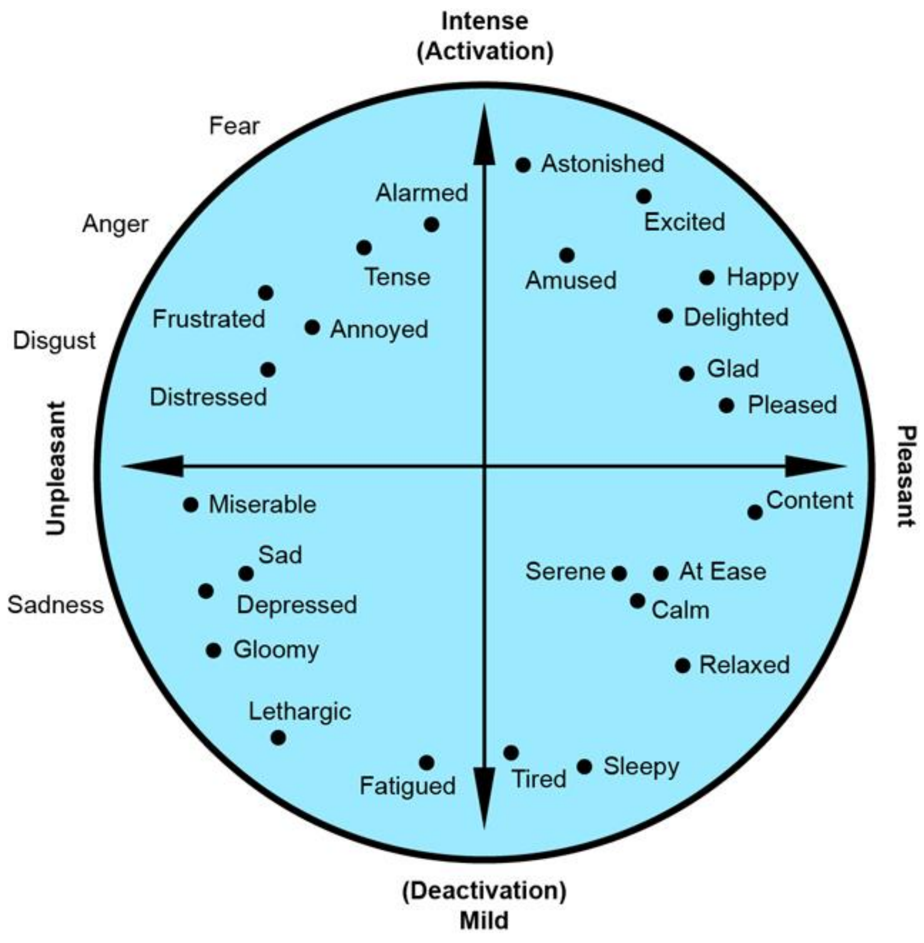
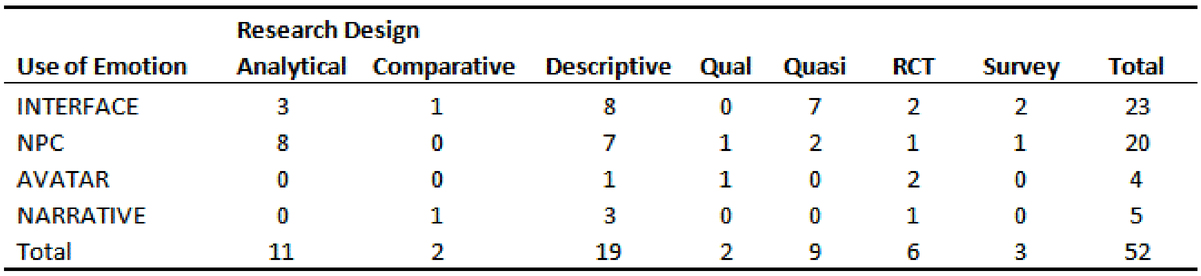


Figure 10: Russell's model of emotions

Emotional modelling has been used in attempts to capture the emotions of NPCs to improve their fidelity and exhibit more realistic behaviours. The difficulty with these approaches is that they are specific to NPCs, they limit approaches to a few emotions and often use simple n-ary vectors to calculate these emotions. While this approach is helpful for NPCs it is untenable for modelling player emotions and attempting to represent emotional changes in this space.

2.4 Emotions and Narrative in Games

Emotions in games have been explored before, however the work is somewhat limited. Indeed, much work on videogames and emotion concerns the reaction of players to various games and research into the link between aggression and videogames [19]. De Byl did a survey of papers that considered the use of emotions in game design [20]. The results can be seen in Figure 11. It is clear there is little in the way of research for both narratives and avatars. The total number of papers is quite small indicating how little papers consider this area in general. The lack of existing work was further motivation for my own research into this overlooked area.



	Research Design							
Use of Emotion	Analytical	Comparative	Descriptive	Qual	Quasi	RCT	Survey	Total
INTERFACE	3	1	8	0	7	2	2	23
NPC	8	0	7	1	2	1	1	20
AVATAR	0	0	1	1	0	2	0	4
NARRATIVE	0	1	3	0	0	1	0	5
Total	11	2	19	2	9	6	3	52

Figure 11: Number of papers focusing on types of emotion use in game design by research design, De Byl

There has been work into the handcrafted storytelling in games and using that to shape the emotions of the player. David Freeman wrote extensively about 'emotioneering' in games, how to craft narratives that would engage players on an emotional level [21]. He advocated for more games to use emotioneering techniques and laid out multiple methods and hypothetical case studies for these techniques. His work make it clear that there are common patterns that can be used to alter the audience's emotional state.

One of the papers that De Byl examined was based upon procedurally generated music in games [22]. The paper described a technique for capturing the emotion of the player in the game. The challenge of the level combined with the player's frustration, and happiness, would be gauged. This in turn would be used to generate music dynamically for the player. This was very encouraging because it follows a similar vein to this work. It attempts to capture a player's emotional state based upon activities in the game. It attempts to use that knowledge to procedurally create content, in this case music.

Affective computing (the use of affects in computing) has been considered before [23]. Picard considered imbuing computers with the ability to express or recognise emotions (though not necessarily *feel* emotions). Picard considered the uses for affective computing in entertainment, film and video, expression etc. Although this dissertation considers a different aspect of affective computing (the evocation of emotions in others), Picard did touch upon an interesting ethical quandary. Picard wondered if affective computing was an area "better left unexplored by humankind". He recognises that the worst that could come of this is emotional manipulation for malicious purposes (a reasonable concern). His response is that this already commonplace (in media, marketing etc.) and so we are better off using computers to understand it. The term 'emotioneering' may seem dubious but the simple fact is that artistic efforts constantly attempt to alter the emotional states of their audience. Comedy, for example, is centred on eliciting laughter and humour but it is not demonised for emotional manipulation.

Chapter 3 Design

Having established the background work to this paper this chapter will discuss the design of the planned system. The aim here is to create a PCG engine that will accurately model the emotions of the player and the emotions they will experience. This will be utilised to change the player's emotional state to some desired goal and to control their emotional experience to achieve a larger narrative.

3.1 The Emotional Model

Firstly, an emotional model is required in order to base any behaviour or content creation on the player's emotional state. As has been discussed emotions are a complex subject and there exist many models for them. Having examined the work in this area it was decided that Russell's model for emotions would be used in the PCG system.

There were several reasons behind this choice. Firstly, Russell's model captures a wide array of emotions. Whilst most models focus on a few emotions or only give a few examples of how any given emotion would be represented in their scheme, this is not the case with Russell's model. In Russell's model emotions are described in terms of two separate aspects. Moreover, many diagrams enumerate the most common emotions, this is clearly demonstrated in figure 12. It is unlikely (but possible) that an emotion based PCG system would want to capture every emotion. Obscure emotions such as bemusement probably will not be particularly sought after. As opposed to models such as Millenson's which isolates three emotions at the expense of the rest, Russell's model leaves a swathe of options open to the developers. Added to this is the fact that the two axes are relatively straightforward. The valence of an emotion, how good or bad it is, is easy enough to gauge. At the very least it can be estimated on, say, a Likert scale. The activation or arousal of an emotion is somewhat more intricate but again it is easy enough to gauge in rough terms.

This makes Russell’s model good for the developer using it to estimate the effect of a scene on the emotional state in this model. It also allows any emotions not explicitly stated on this model to be placed, roughly on this model. Therefore, more emotions could be plotted, leaving the entire emotional spectrum open to representation. As a bonus, using a two-element vector for emotional states saves on memory costs by the PCG system.

The popular method for modelling emotions in industry is mostly centred on NPCs which have limited behaviours by necessity. Therefore, their emotional range is restricted to several key emotions. This is useful for NPCs but results in limited expressive power. As previously discussed the superior expressive power of Russell’s model is more appealing.

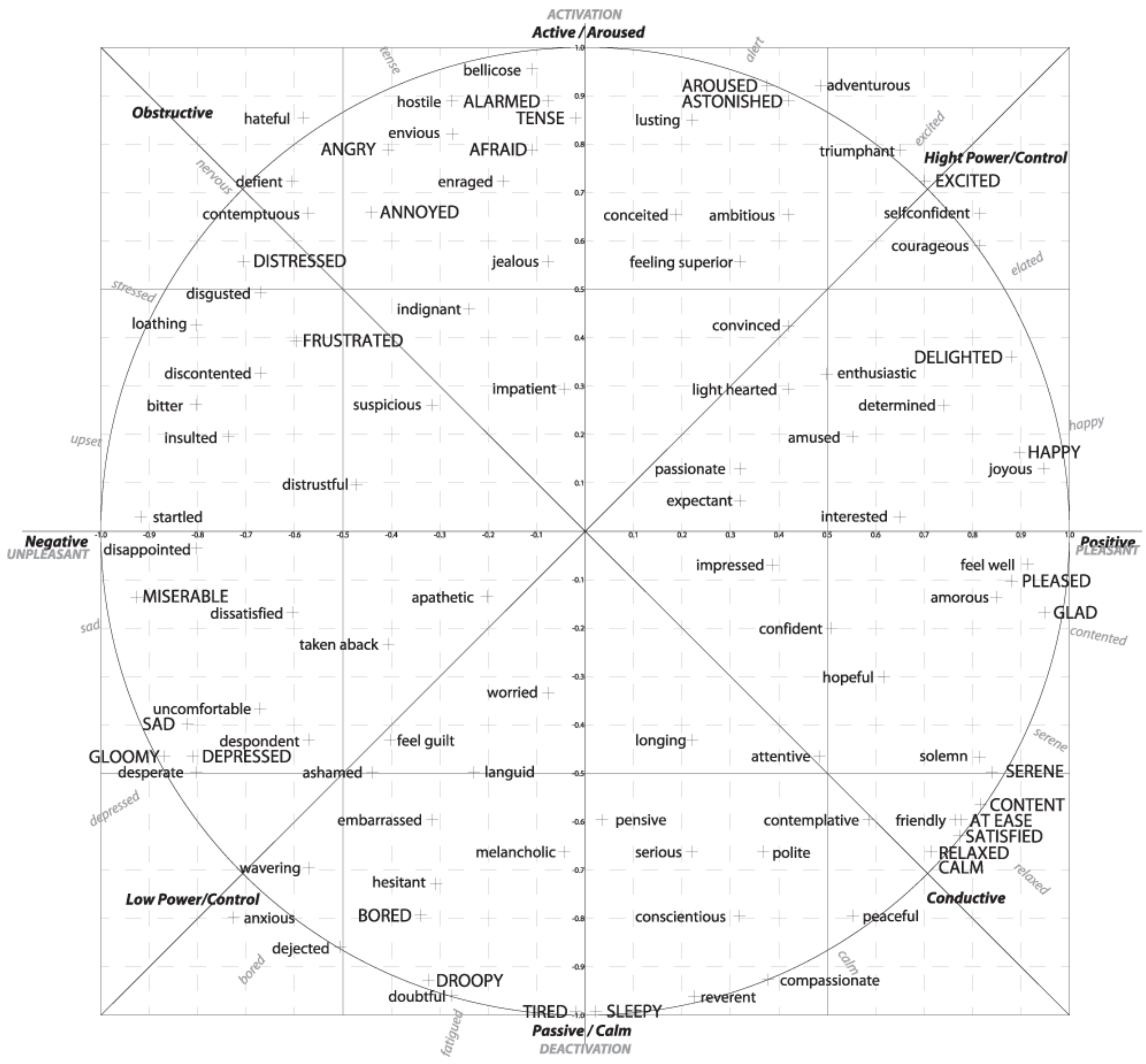


Figure 12: More detailed view of Russell's model

In the *Fable* games NPCs used vector states to model their emotions. This is used to determine their reaction to the player e.g. if they are afraid of or happy around the player. This requires actions to apply modifiers to five element vectors. Similarly, an attempt to capture even four or five emotions would require: the player's state to be stored in this form, and to have actions apply vectors in this space. While this could turn into ten, or twelve vectors even with only four elements it is not as attractive. What do these emotions actually represent? And the change vectors applied? It is far easier to conceptualise Russell's model. For one, there are diagrams to aid with comprehending it. As described it is easy enough to gauge where an emotion would lie in this model. More importantly if the designer knows what they want the player to feel and can express this verbally, it is likely they can describe this as a vector on Russell's model.

Diagrams such as Plutchik's wheel are nice for exploring emotions in psychological terms. But for representing emotional states numerically and changes therein they leave much to be desired.

With all this considered, Russell's model was the obvious choice for this research. It has expressive power, an intuitive relation to the real world and can be easily modelled in a game engine as a two-element vector. For the purpose of this research emotional states will be a two-element floating point vector in the space of this model. It is assumed that the state is limited to the range (-1, 1) on both axes. While the diagrams do imply a circular model, I did not think it necessary to apply this restriction, but it easily could be constrained to the unit vector with minor adjustments.

3.2 The Augmented Grammar

With the emotional model selected, the PCG system must be updated to use it. This dissertation is concerned with the emotional guided generation of content, not the mechanics behind that generation itself. For this reason, De Kegel's puzzle system was used as the basis for this system. This allowed significant code reuse, most importantly the game engine created for that system could be used for this research. This gave the groundwork for the emotional model to be added. The key to the emotional model was keeping it simple, having chosen Russell's model few changes were required to the overall PCG system. The first, obvious change was to give the player class a new variable to track their emotional state. This two-vector variable is used to track their current state in Russell's model at any given time.

This new state had to be altered, and altered at the appropriate juncture. The context-free grammar used by the puzzle generator gave the ideal moment to alter the emotional state. The player's emotional state can be expected to change when something happens, when an action is taken. In the meantime, nothing is happening (from the perspective of the game engine). For this reason, the grammar was augmented, the major changes were to the rules used. Each rule in the database represents the player taking some action, this is the most significant time for the player to experience some emotional change (aside from some automated event in the narrative, which could be handled elsewhere). Therefore, the Rule class was given its own two-element vector, not to represent an emotional state, but to represent a change in the emotional state. Each rule will supply some change vector to the player's emotional state in Russell's model. Whenever the rule is applied during the game's runtime the emotional change will be used to alter the player's emotional state.

The new and augmented grammar is depicted in Figure 13 below. The rules must now incorporate the state change vector.

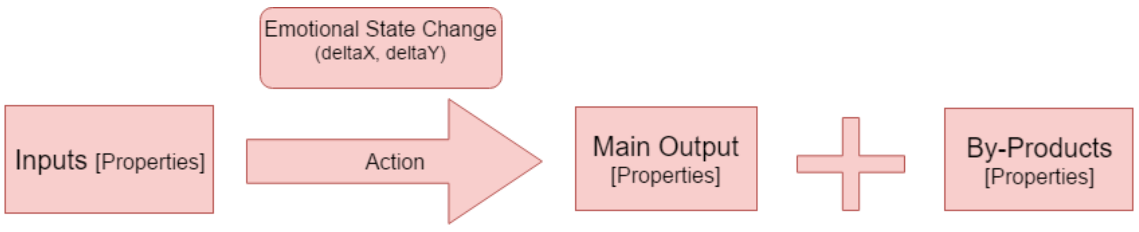


Figure 13: Augmented Grammar rule, includes state change vector

The augmented grammar provides a means to alter the player's state but the rules must still be selected to produce a desired affect. Random rules would provide no cohesion to a larger emotional narrative. For this reason, the Area class is also updated. The game area determines the goal that a player must achieve to progress. It was sensible therefore to relegate the desired state variable to this class. The area will track the restrictions placed upon the puzzles generated for that area. This gives the engine the desired state for the player to end in. This information is used in turn in the search system, explained in greater detail below.

Below are some sample rules that could be used within the PCG system (figure 14). Each one shows the inputs and action taken leading to the outputs. Additionally, there is now an emotional tag associated with each action that changes the player's emotional state.

$$\boldsymbol{Tree} + \boldsymbol{Axe} \xrightarrow{\boldsymbol{Chop}_{(0.1,\ 0.1)}} \boldsymbol{Wood} + \boldsymbol{Axe}$$

$$\boldsymbol{Chest}_{\boldsymbol{Locked}} + \boldsymbol{Key} \xrightarrow{\boldsymbol{Unlock}_{(0.2,\ 0.1)}} \boldsymbol{Chest}_{\boldsymbol{Unlocked}} + \boldsymbol{Key}$$

$$\boldsymbol{Door}_{\boldsymbol{Locked}} + \boldsymbol{Axe} \xrightarrow{\boldsymbol{Break}_{(-0.1,\ 0.2)}} \boldsymbol{Door}_{\boldsymbol{Broken}}$$

Figure 14: Sample rules for the PCG system, with emotional tags

3.3 The Puzzle Creation System

With the emotional model used to augment the grammar the system can model emotional states and the changes in those states. As before the game consists of numerous areas that are predefined by the game developer or designer. Areas are connected to each other allowing scenes to be sequenced together. Each area now has its completion goal and the emotional state, this sets how the player should feel after completing that area. This allows a degree of pacing to be enforced by the developers which can follow structures such as Freytag's pyramid or more intricate narratives. Alternatively, the developers could specify certain sequences of emotions, varying from horror and tension to relief as the conclusion draws near. In this way, the developer still maintains a lot of control over the narrative but there is still variety in the experience due to the various methods of evoking the required emotions as given by the rules.

Just like in the original system a puzzle tree is created for each area, which determines the puzzle that the player will encounter. The difference is that the emotional tags added to the rules will alter what puzzle is selected. Unlike before where rules were selected at random (as they made no conceptual difference to the end result) the rules are now guided by a search process.

3.4 The Search System

In order for the player to experience emotional change they must go from one state to another. This requires their starting state, or the current state they are in and the state they must end in. As mentioned above, this is dictated by the current area they are in. The puzzle tree selected for the area must be chosen so that it achieves the local, area goal. More importantly it must now be selected so that the rules applied will change the player's emotional state to the desired state. This requires a search function that will satisfy these criteria.

The puzzle tree is created by working backwards from the goal to a certain depth (setting the difficulty of the puzzle). The focus for this system is not the depth but the emotional state of the player. The rule database determines each of the different rules that could be applied to yield the required goal(s). These rules will also now contain a change to the player's emotional state. Therefore, each puzzle tree will have a cumulative effect on the player's emotional state from what it currently is. This will determine the emotional change they experience in this area. The search for the correct puzzle tree will need to use a search algorithm over this space. This operation will only be run once, before the area is populated. For this reason, I think a form of breadth-first search (BFS) would be best. Tracking a list of the possible puzzle trees, and the emotional change for each one. At each stage the puzzle tree closest to the goal will be selected and expanded (based on the rules that could be applied). If no tree would result in the correct emotional change then the system can either select the closest state change or end with an error (a poor decision for a released game but suitable for development). Either way the failure should be reported to the developer.

Chapter 4

Implementation

This chapter discusses the implementation of the system described above. The system was built in Unity using C# scripts. As explained above the work was building on the original puzzle generator created by De Kegel and so most of the work was in simply adding and altering to this original code.

The puzzle generator uses Unity assets to populate the game world and to provide the game engine. The C# scripts themselves provide the implementations for the context-free grammar and the related terms, rules etc. Unity prefabs, game objects whose configuration and properties are saved for reuse, are used to provide the game objects.

Unity is a game engine that is free to use for educational and non-commercial purposes. It is a popular game development tool due to its accessibility, cross-platform support and large community support. Unity is capable of exporting games to most game platforms and has a store for obtaining further game assets (with many being available for free). Unity is an ideal choice for exploring this concept because it avoids the need to create a custom game engine and is widely accessible to game developers and even those new to Unity.

4.1 The Game World

The game world itself consists of a three-dimensional world filled with scenery such as mountains, trees etc. The player is restricted movement on the 2D ground plane. Unity provides the physics and rendering for the game. The world is prepopulated with items adding to the scenery and with limited player interactions. Any items required for the puzzle that are not present in the scene are automatically added using several pre-set spawn points which must be added to the scene by the developer. The scene can be viewed in figure 15. The game is only there for testing the content generation. In an industry title, more sophisticated game assets would be used for higher production value.

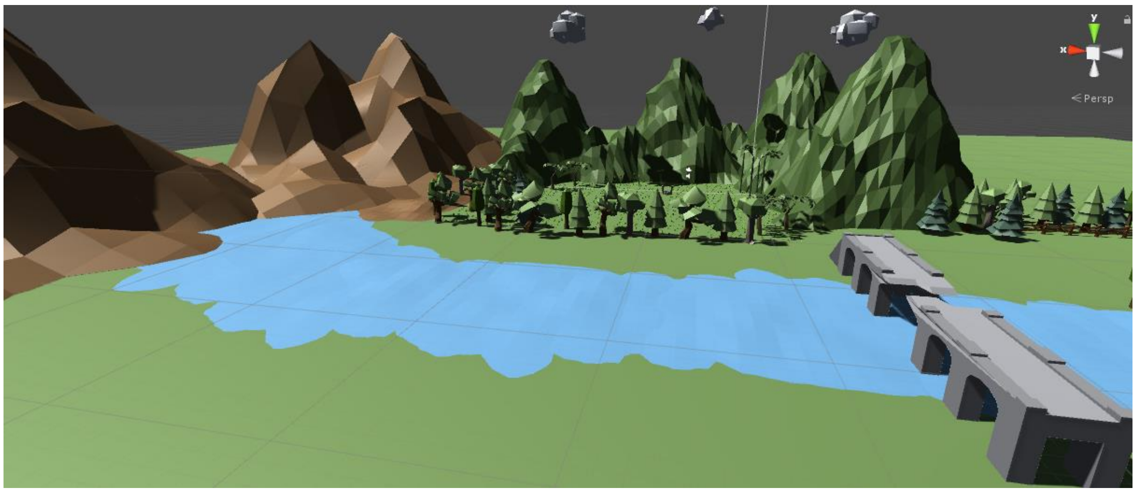


Figure 15: The game world, viewed from the Unity editor

The game only contains two areas, and thus only two puzzle trees are generated. To add further areas the Unity scene would need to be developed further allowing for a longer emotional journey to be portrayed. These new areas would also need to be developed with appropriate scenery and spawn points added.

4.2 The Puzzle Generator

The changes introduced were mostly confined to the C# scripts because they do the heavy lifting for the puzzle generator. The game area and player classes were augmented with their new emotional state vectors. The player class was given functions to alter the state and ensure that it remains in the range (-1, 1) on both axes. While this could be enforced elsewhere it is unnecessary because it is currently only used for the player and must only be checked when the state is altered.

The rules were changed to include their new vector for state changes. The application of rules was changed to add this vector to the player's emotional state vector. The largest change however was to the puzzle generator itself, as would be expected. The process of selecting rules from the database was altered from a random selection to one based upon the emotional states of the rule.

The selection process for generating puzzles uses a recursive method for breaking down the inputs to each rule until the puzzle cannot be broken down further or a desired complexity is reached. This recursive nature made it difficult to implement the BFS style search envisaged in the design of this system. The new puzzle generator must now consider and track all the possible puzzle trees available to find one that ends in the correct state. This difficulty prevented full implementation of the system and required a simpler approach to searching through the emotional space. This is described in greater detail in chapter 5.

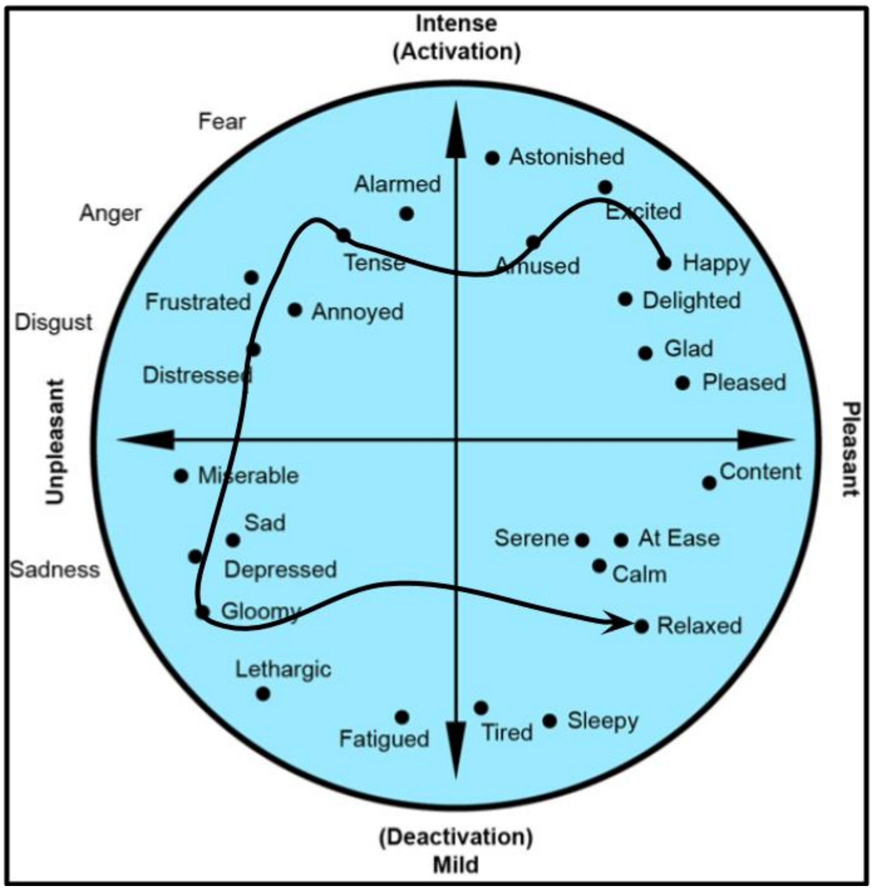


Figure 16: Example of a path through the emotional state space

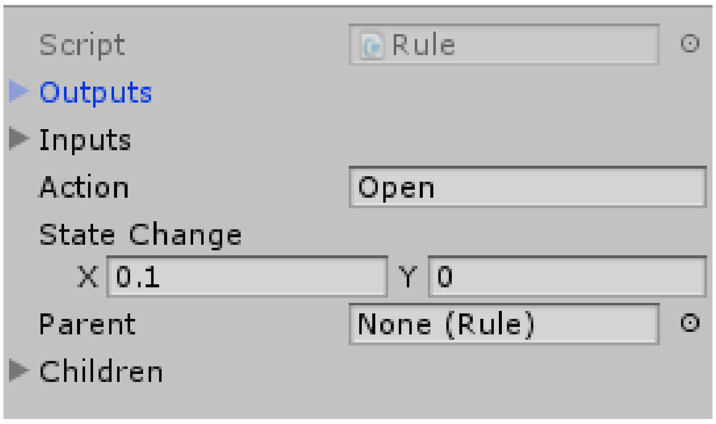
The resulting puzzle generator creates a path through the emotional model for each area. Combined in sequence the effect creates a much larger path through the model which gives structure to the puzzles. This can be visualised in figure 16 where a possible path through the model is given. This best expresses the potential control provided by the augmented grammar. By specifying certain key emotional beats in each area, the designer can create a path similar to this. The main difference and advantage is that PCG enables the destination to remain the same while the journey changes from each implementation.

4.3 The Unity Editor

For the game developers that would use this system the knowledge of the underlying system implementation would not be necessary. Their concern would be the rules and items that populate the game world. Much of this is unnecessary for this dissertation however the important aspect of creating a game with this system is the use of rules within the game world and their associated affect on the player's emotional state.

As has been stated the rules compose the actions that the player can take, it is in the Unity Editor that they can be created, edited and deleted. The interface provides easy access to the terms that constitute a rule and thus their effect on the game. By editing these the designers choose the actions allowed in their game, and most importantly the effect on the emotional state. The view of the editor is shown in figure 17, below.

New to the system is the state change variable which is attached to the C# object. Unity helpfully links to the C# scripts and displays all the variables which can be edited as required. Here is where the developers would alter the effect of the rules and in the process change when each puzzle would appear and how they expect them to affect the player.



Script	Rule
▶ Outputs	
▶ Inputs	
Action	Open
State Change	
X 0.1	Y 0
Parent	None (Rule)
▶ Children	

Figure 17: View of the rule editing tool in Unity, now the rule includes a state change

When implementing my changes, it was necessary to edit the existing rules provided by the original code base. The rules were updated with their own state change values which were based on my own estimates for their affect on the player with some slight exaggeration for the purposes of this proof of concept.

Chapter 5

Evaluation

The system created will be examined in this section and its fulfilment of the objectives outlined in chapter 1 will be explored. The merits of the system created are reviewed and the cost of implementation evaluated. The augmented system is also examined in relation to the previous puzzle system.

5.1 Game Demo

The system implemented uses Unity for the game environment with the puzzle generator scripts handling the emotional model. The Unity editor is used to tweak the emotional state variables for rules and the areas. Although the player can be initialised to some emotional state it is safest to assume they are in the neutral (0, 0) state. As with the original puzzle system the scene is populated with two areas with some simple starting objectives. There was no need to change the game mechanics or items in the scene beyond adding the emotional tags. In the first area, the player must acquire an axe in order to get out of their starting area. This either requires them to help a sick animal or solve a puzzle to open a chest. It is estimated that the player will respond positively to these events (by helping a sick creature and solving the puzzle).

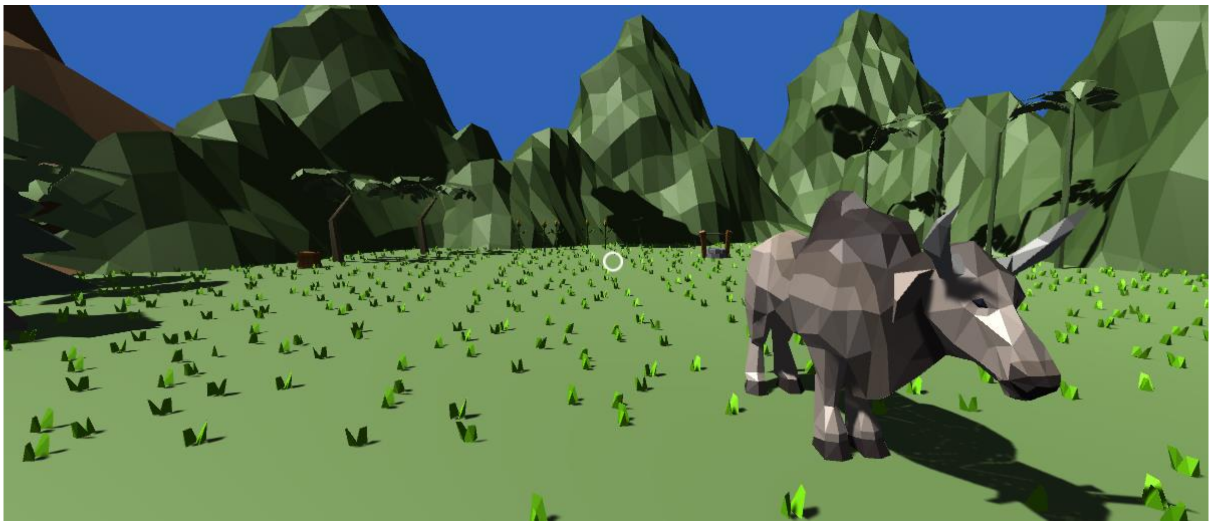


Figure 18: Snapshot from the game engine

The puzzle generator will select from these puzzle trees to get the player to the target state for the first area, which is set to (0.5, 0.3). After the player completes these tasks the game would expect the player to be in that state, and then to proceed to the next area which would have its own goal state. In the same process a puzzle tree is selected to achieve this change to the new goal state. In this way, each area selects a puzzle tree that combines to move the player closer to the desired state. If there is no transition to the goal state, the system will continue with the procedural generation (so that the game can continue) but will flag that the generator did not move to the desired state.

The game segment demonstrated is unfortunately small, especially when compared to a commercial game. This was a disappointing reality of the research. It is not feasible to create the same large scale game that matches a professional endeavour. With PCG it is easier to create such a game but an emotionally diverse narrative requires significant game assets and development time just to create a simple test. Significant efforts would need to be placed in creating scenarios that can affect the player emotionally, which could warrant their own research.

Implementation difficulties resulted in a rudimentary search system in the emotional space. The design put forth in chapter 3 would intelligently search through the possible puzzle trees to find the one which would yield the desired state for an area or at least the closest state possible in the emotional space. The previous system was centred on guaranteeing the solvability of the puzzles and rightly so. However, this led to a focus on selecting a puzzle tree with the correct depth and selecting a possible rule to apply at random to create the tree. Due to technical difficulties, I was unable to reconfigure this search to consider the various possible trees and select the best candidate in time. It certainly is possible but is not present in this implementation. Instead the system takes a greedy solution, at each stage selecting the rule that brings it the closest to the desired state immediately, this may lead it to local maxima and missing correct paths. This is a small and fixable problem, in the resulting system, the overall concept remains present.

The main outcome here is the design of an emotionally focused PCG system by combing PCG puzzle engines with the emotional model to create a narrative based experience for the player. One of the important results here is that there is work that has yet to be done, there is no dead end in this area and the system is versatile enough that if changes are necessary they could be made.

5.2 Using the Model

The model still requires human intervention and maintenance. The initial environments must be created, although using another PCG system these could be created dynamically. Most importantly the game developers or designers must instantiate each rule individually and determine its change to the player's state. It is likely that most actions would be quite small and result in little if any change to the player's state. In this proof of concept the expected changes were exaggerated for demonstration purposes.

The system itself is easily accessible, rules can be added and altered on the fly allowing for adjustments to be made or for continuous addition of content which would make the puzzles more diverse. While this implementation is based on the puzzle generator from De Kegel's work there is very little tying the system to this PCG engine. The core concept it relies upon is the emotional change in particular actions or key events. The concepts could be applied to another system and in fact this could remove insignificant state changes from small actions which don't cause an affect in the player.

The narrative in this system is divided into the particular areas the player encounters. Each area can be altered and depending on the size and frequency of this sections the developers can choose how much control they have over the emotional changes. A few sections would create a rough path for the player to follow, while dozens of levels would set a very specific journey for the player.

5.3 The Emotional Model

Using Russell's model for emotions proved to be a good decision. Firstly, the representation is small and concise which makes it easier for the developers to use and understand. As has been noted a common measure for player modelling considers the tension or engagement the player has. It is very common for games to have challenging sections mixed with calmer, easier segments. This is captured by one of the axes in Russell's model and so is an easy concept to understand. Intense segments of the game can be expected to move the player further up this axis on the model.

In this implementation, the puzzle system takes a direct path between emotional states. This was for simplicity but the system can be changed by changing the search system. For example: the engine could look for a curve to get from state A to state B or a staggered, zig-zag pattern. The emotional model is fixed into the implementation but the search process is kept separate and could be updated or replaced. The system is versatile to change and could support several options for operating each with varying effects.

5.4 Comparison with Previous System

Implementing the emotional model has enhanced the capacity of the previous grammar without detracting from it. All the features and capabilities of the previous system remain. It is even possible to use the original system by simply setting the emotional changes correctly. As with the existing system the puzzles are all solvable. The emotional model looks at a more abstract view than the puzzle system and would possibly benefit from a different base engine focusing on character interactions and other narrative heavy elements. The concepts themselves remain sound but now the puzzles emphasise an emotional change.

Compared with the Puzzle-Dice system this has a different focus but it is still emphasising dynamic narratives over empty content. In the system created there is greater emphasis on the metadata about rules and items. In the basic puzzle generation systems, the engines relied upon the physical properties of the game items within the game. One of the big changes for the emotional based system is that game items must have meta properties which also begins to lead towards NPCs. In the most successful commercial narratives NPCs play a substantial role, it would be more effective if a PCG engine could utilise this to create more compelling emotional experiences. Grief, betrayal, relief and anger are much easier to elicit when the player has someone to connect to, even a virtual character or animal.

Chapter 6

Conclusion

This chapter will briefly conclude this dissertation with an evaluation of the work and some notes on the research limitations. Some possible ideas for future work will be discussed and some final remarks on this project will be given.

6.1 Main Contributions

With the Puzzle-Dice system and the work of De Kegel there have has been progress on creating puzzle systems through PCG for narrative purposes. The work here aims to augment those systems with a more direct approach to creating affect within the players. This is to create more engaging narrative stories and to allow better artistic direction over the PCG systems used today. This work builds upon the existing narrative puzzle system designed by De Kegel by combining it with an emotional model based on Russell's work. By tracking the emotional state of the player and guiding it through dynamic content more engaging experiences can be created.

The system written in Unity by De Kegel was expanded upon to incorporate this emotional model. The puzzle generation system was altered to ensure both the solvability of the puzzle and the desired change in the player's emotional state. This is a small demonstration of the possible system that could be used in industry.

6.2 Limitations

The system described here does have some limitations which do impede its research impact. This concept is uncharted territory and does not have the same quantitative measures that other work in this area shares. The ability to affect the player with this content is difficult to measure.

First and foremost, there is a distinct variety in people, and this makes emotional modelling difficult. It is hard to know exactly how a person will react to a given scene in a game. It is impossible to guarantee the same reaction from every person.

Additionally, the starting state of the player may vary wildly from one session to another as the player starts a game with an emotional state that cannot be measured.

Although there are tried and tested methods that can be used for eliciting emotions there is still an onus on the game developer to understand their audience and to craft puzzles and scenes that can cause the desired emotional affect. The developers would want to test particular puzzles to check that they are exhibiting the expected behaviour and adjust the puzzle or the emotional tags as necessary.

The search method employed to get the player from one emotional state to the other could be improved and expanded upon. The terms used in the puzzle generation system would benefit from metadata that could provide insight for the puzzle generation system e.g. the fact that a player often uses a specific tool could indicate emotional attachment or at least indicate a way to induce emotional affect by altering this item for better or worse.

The starkest limitation of this work is its scope. Unfortunately, there was insufficient time and resources to adequately test this system. The creation of a small game with numerous rules and items would enable case studies and the system could be more accurately judged on its ability to create strong narrative experiences.

6.3 Future Work

It is clear to me that this area should be explored further. If PCG is to have any real narrative impact it must be able to alter the story and emotional engagement of the player. While this presents some implementation difficulties it is not necessarily impossible. The central question that I ask is: ‘Can PCG be used for story elements that affect the player emotionally?’. This is difficult to quantify and indeed test. Given the capabilities of developers to emotionally invest players in their stories this does not seem out of reach. Combining this with the structural nature of narrative and the emotional model presented in this dissertation this certainly seems feasible.

With the method to correctly evoke emotional responses from the players the next step would be to tie together sequences of these narrative puzzles. This could be done using some artificial director which varies the emotions based on previous scenarios so as not to repeat the same note. On top of this it is almost certain that the player will experience emotions even when action rules are not being applied. A static function which alters the player’s state over time could be used to model this. For example: if a player is just exploring it is likely any strong emotions they have will mellow as time goes on (while they are not subjected to any major stimulus).

This leaves a significant task in finding and determining which emotions to evoke and how this is possible. In many industry examples the compelling stories come not just from the environment but also from the characters that the player interacts with. Creating such content procedurally is certainly a difficult task but it would increase the ability to affect the player by increasing their connection to the narrative that is being explored.

There is much work to be done in this area but there is much to be gained from it. Expanding on the work in this dissertation I would recommend fleshing out the content used by this puzzle generator. Creating a more substantial demonstration with more content would enable tests to be carried out to get feedback from players. The most critical aspect of this work is the viability as a development tool which presupposes that it can accurately ‘emotioneer’ the players. This also requires further work into finding those scenes which induce an emotional affect in the player.

Furthermore, the puzzle generator itself could be built upon. A more detailed search function when finding the sequence of puzzles to give the player could create more interesting affects. Additionally, I think that more abstraction would benefit the engine. To track information such as: how many times a puzzle has been used, or what characters the player interacts most with. This system aims to avoid repetition, so avoiding tasks the player has already done numerous times would be a logical addition. For example: if a player interacts with a single character they are likely to grow bored with them unless they can be involved in a new story or puzzle in an interesting way. It would also enable the system to gauge how invested the player is in individual locations, characters, and items.

It occurred to me that the emotional modelling used in this system could also be used in other aspects of videogame development. For example: most emotional models in the industry aim to improve the fidelity of non-player characters. Similar to the player modelling; this system could model the NPC's emotional states using Russell's system.

This could be improved further by having player actions (the application of rules) applied alter the state of the NPCs and the player. E.g. Helping a character may improve their state and change their behaviour, while harming them may make them angry, or scared and cause different reactions.

6.4 Final Thoughts

In summary, there is a host of options for expanding this work and not many people are exploring this aspect of emotions or PCG. I recognise that this work has its limits but I believe it to be a system work exploring in greater detail. There are sufficient applications for this work in industry and in academia. Modern computational power can create incredibly rich and beautiful environments automatically, and it is possible to populate those worlds for a more powerful and lasting experience that will remain with the player long after they stop playing the game.

Bibliography

[1] M. &. W. G. Toy, *Rogue,* Epyx, 1980.

[2] T. Adams, *Dwarf Fortress,* Bay 12 Games, 2006.

[3] Mojang, *Minecraft,* Stockholm: Mojang, 2011.

[4] Hello Games, *No Man's Sky,* Hello Games, 2016.

[5] T. Rychnovsky, “Procedural Generation of Puzzle Game Levels,” 2014. [Online]. Available: http://www.gamedev.net/page/resources/_/technical/game-programming/procedural-generation-of-puzzle-game-levels-r3862.

[6] S. Colton, “Automated puzzle generation,” in *Proceedings of the AISB'02 Symposium on AI and Creativity in the Arts and Science*, London, 2002.

[7] Bethesda Game Studios, *The Elder Scrolls V: Skyrim,* Bethesda Softworks, 2011.

[8] B. De Kegel, “Procedural Generation of Narrative Puzzles,” Dublin, 2016.

[9] C. &. T. A. Fernández-Vara, “Procedural Generation of Narrative Puzzles,” Massachusetts Institute of Technology, 2012.

[10] P. Gorman, “Dramatic Structure,” [Online].

[11] J. Wesołowski, “Beyond Pacing: Games Aren't Hollywood,” 21 May 2009. [Online]. Available: http://www.gamasutra.com/view/feature/4032/beyond_pacing_games_arent_.php?print=1.

[12] M. a. S. A. Mateas, *Façade,* 2005.

[13] Turtle Rock Studios, *Left 4 Dead,* Valve Corporation, 2008.

[14] B. &. R. J. A. Fehr, “Concept of emotion viewed from a prototype perspective,” *Journal of Experimental Psychology: General,* pp. 113, 464-486, 1984.

[15] J. R. Millenson, The psychology of emotion: Theories of emotion perspective, John Wiley & Sons, 1967.

[16] J. R. Millenson, Principles of Behavioural Analysis, New York: Macmillan, 1967.

[17] A. e. a. Ayesh, “A Millenson-Based Approach to Emotion Modelling,” *Centre for Computational Intelligence,* 2008.

[18] J. A. Russell, “A Circumplex Model of Affect,” *Journal of personality and social psychology,* 1980.

[19] N. e. a. Ravaya, “Emotional Response Patterns and Sense of Presence during Video Games: Potential Criterion Variables for Game Design,” Tampere, Finland, 2004.

[20] P. De Byl, “A conceptual affective design framework for the use of emotions in computer game design,” *Cyberpsychology: Journal of Psychosocial Research on Cyberspace,* 2015.

[21] D. Freeman, “Creating emotion in games: The craft and art of emotioneering,” *ACM Computers in Entertainment,* pp. 15-15, 2004.

[22] D. &. P. D. Morelli, “Experience-Driven Procedural Music Generation for Games,” *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 4, NO. 3.,* 2012.

[23] R. W. Picard, “Affective computing,” MIT press., 1997.