

# Diage: A Dialogue Generator

Nico Andrew Glas

11th February 2014



# Abstract

*will come later*[1] [3] [4] [8] [14] [11] [10] [12] [13]



# Acknowledgements



# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Create-IT . . . . .	9
1.2	Technology . . . . .	9
1.2.1	Diage . . . . .	9
1.2.2	Rouge . . . . .	9
1.3	Research Phases . . . . .	10
1.4	Problem definition . . . . .	10
1.5	Research methods . . . . .	10
1.5.1	What gameplay benefits can we introduce? . . . . .	10
1.5.2	How does the development process benefit from a procedurally generated narrative? . . . . .	11
1.6	Proof of concept . . . . .	11
1.7	Requirements and Constraints . . . . .	11
<b>2</b>	<b>Procedural Content Generation</b>	<b>13</b>
2.1	Narrative . . . . .	13
2.2	Static and dynamic generation . . . . .	14
<b>3</b>	<b>The interactive story</b>	<b>15</b>
3.1	Video Games and Interactive Stories . . . . .	15
3.2	IS . . . . .	15
3.3	Dynamic plot generation . . . . .	16
3.4	Character-based storytelling . . . . .	16
3.5	Riedl and Young . . . . .	16
3.6	Generative narrative within games . . . . .	16
3.7	Conclusions . . . . .	18
<b>4</b>	<b>Narrative planning</b>	<b>19</b>
4.1	Actor attributes . . . . .	19
4.2	Initial Generation . . . . .	19
4.3	Step Generation . . . . .	22
4.4	Events . . . . .	22
4.5	Planning flow . . . . .	23
4.6	Narrative planning in Rouge . . . . .	23
<b>5</b>	<b>Diage Modelling Language</b>	<b>25</b>
5.1	Entities . . . . .	25
5.1.1	Objects . . . . .	25
5.1.2	Actors . . . . .	26
5.2	Information Containers . . . . .	26
5.2.1	Spaces . . . . .	26
5.2.2	Events . . . . .	26

5.3	Actions and Accessors . . . . .	27
5.4	DML Creator . . . . .	28
<b>6</b>	<b>Rouge</b>	<b>29</b>
6.1	Mechanics . . . . .	29
6.2	Cellular Automata . . . . .	29
6.3	Flood filling spaces . . . . .	30
<b>7</b>	<b>Conclusion</b>	<b>33</b>
7.1	Further Study . . . . .	33
<b>8</b>	<b>Reflection</b>	<b>35</b>
8.1	Diage . . . . .	35
8.2	The literature . . . . .	35



# Chapter 1

## Introduction

During my coursework for my Bachelors degree in Information Technology, I stumbled upon the field of procedural content generation (pcg). I dedicated last two years of school to understanding and applying this field, and it silently turned into one of my 'specialities'. It all started with a research project that focussed on generating game worlds with Voronoi diagrams. After that I made a couple of games that used content generation for various things; from using audio to generate platforms for an *endless runner* type game, to cellular automata to create dungeons for a *dungeon crawler*. The one thing I always had an interest in, is video game narrative. My thoughts turned towards the idea of using procedural content generation techniques to create a narrative for video games. Together with the Joris Dormans of the Create-IT research facility we set up a project to research the use of pcg in video game storytelling. This thesis is presented as part of my fulfilment for graduation and serves as a research report.

### 1.1 Create-IT

Create-IT applied research is one of the research institutes hosted by the University of Applied Sciences of Amsterdam. In this lab students, teachers and researchers all preform applied studies in the different sections of the IT world. Their goal is to educate future professionals in the uses of applied research, so these professionals can anticipate the ever changing field that is Information Technology. In the newly created Game Research Lab students and researchers alike contribute to the growing community of game developers; making the development of games easier or trying to understand current problems within the industry.

### 1.2 Technology

This section covers the technologies used by me in developing and demonstrating the *Diage* system.

#### 1.2.1 Diage

General Programming	C#.Net
Graphing	Gliffy
Prototyping	Ludoscope

#### 1.2.2 Rouge

Programming	C#.Net
Game Engine	SilicaLib
Framework	XNA

## 1.3 Research Phases

This project lasts for 20 weeks, and the following will indicate the phases of my research process.

- **Week 1-6** Literary study
- **Week 7-13** Creating a generative algorithm
- **Week 14-20** Proof of concept.

## 1.4 Problem definition

My main research question is *How can a roguelike game benefit from a procedurally generated narrative?*. In this section I will state my research questions, and try to dissect them so all readers of this document have the same definitions, and the same context.

Let me clarify the term *roguelike*. The genre started with the video game **Rogue** that was released in 1980, and was characterised by having "random" dungeons where the player has to navigate rooms and fight monsters. The ultimate goal of the game was to get to the highest level possible without dying once. The game never really "ended". The game was over when the player died, but after that the player got to start all over again on level 1 with a complete newly generated dungeon. As the game gets progressively harder when the player starts go get to other levels, the chances for the player to lose get higher. Now, back to the term "*roguelike*"; A game with no definitive end and permanent loss of game progress when the player dies. The previous years has seen a rise in popular *roguelike* games, **FTL: Faster Than Light** by *Subset Games* (2012) and **The Binding of Isaac** by *Edmund McMillen and Floris Himsl* (2011) being just some examples.

I specifically target *roguelikes* for their inherent use of content generation. The need to have a different set of content throughout a play-session is the key selling point of a *roguelike* game. This makes it the right genre to experiment in with any new type of content generation.

In my question I speak of benefits to the game by the use of a procedurally generated narrative. This results in a few sub-questions that tackle these benefits. *What gameplay benefits can we introduce*, and *How does the development process benefit from a procedurally generated narrative*. These questions are measurable to a certain degree that give us a good view on the beneficial factor of a generated narrative.

## 1.5 Research methods

This section will cover the methods used to measure the benefits that adding a procedurally generated narrative has.

### 1.5.1 What gameplay benefits can we introduce?

If there ever was a noun which definition was disputed, it's probably *gameplay*. There is, however, one definition that I feel covers most facets: "*The experience of gameplay is one of interacting with a game design in the performance of cognitive tasks, with a variety of emotions arising from or associated with different elements of motivation, task performance and completion.*" (Lindley, et al. 2008). [something about the quote]. I want the generated narrative to be part of a game, not just an additive thereon. With the close ties generated content has with emergent behaviour, I want to explore the changes that get introduced when adding a dynamic narrative to gameplay.

### 1.5.2 How does the development process benefit from a procedurally generated narrative?

This questions has some caveats. What does it take to measure a process? Do we look at time spent writing a story and contrast that with the time spent building a story generator? What about the fact that we only have to build that generator once, whereas story crafting needs to be done for every single game. These questions all get tackled in a later chapter, when I discuss the proof of concept that has arisen from this project.

## 1.6 Proof of concept

As a proof of concept I propose to build a game that incorporates the findings of this research. The game I will make will be a *roguelike* for their inherent use of PCG techniques. Due to the limited duration and scope of this project, the game will be restrained to the most basic elements of a *roguelike*. The only addition being a dynamically generated narrative. This game should pose as the proof of my research and be demonstrable to verify my answers to my research questions.

## 1.7 Requirements and Constraints

The project has several requirements and constraints, which are:

- *Only roguelike*: All research done into generating content is targeted at *roguelike* games. This is done because said genre are small and already rely on procedural content.
- *Windows only*: I will develop only on *Microsoft Windows* for the duration of this project, thus limiting the resulting products to be available only on Windows.
- *Working demo*: The project must result in a working demo that shows the potential of a procedurally generated narrative.



# Chapter 2

## Procedural Content Generation

In the world of game development we strive to create the best experiences for a wide and diverse audience. During the course of development there are a number of obstacles that can (and often will) hinder the progress of the developed game. Some of these hindrances come from processes that are essential to a game like; level- and world building, or story- and quest design. They take a proportionally large amount of development time and take a very specific mindset to create. A current development within the game development<sup>1</sup> community to tackle these issues is the use of *Procedural Content Generation* (PCG). This may be a confusing phrase for some, so let's dissect it.<sup>2</sup>

**Content Generation** When taken apart, this part of the phrase will probably become a lot clearer. It just really means what it says, we try to generate content, be that a quest for a Role-Playing game (like **World of Warcraft**) or a complete dungeon for a dungeon crawler (like **Rogue** or the **Diablo** series), content means anything within the context of a game. This generation can take place either on run-time (sometimes called **live** or **on-line** generation), making it possible for new generation for each time the game is run. Another possibility is to generate content prior to release. This has the benefit of being able to tweak the content to make it more satisfying or aesthetically pleasing, but removes the dynamic aspect that some games might strive for. Nobody will say that one of these methods is *better* than the other, but they are both tools to be used on the right moment.

**Procedural** This part of the phrase is the one that often causes the most confusion, but just means that we use a specific procedure to generate our content. In the world of programming, it usually comes down to one or multiple algorithms that work together to generate *predictable* content. A good example for a non-digital form of procedurally generated content is the card game **Klondike** (also called **Solitaire** or **Patience** in the US and the UK respectively). The procedure in this game is the shuffling of the deck. This "procedure" ensures a random order of the cards, and the resulting content is the layout of the game. While this might be a good example for content generation it is a poorly implemented one, for there exists multiple outcomes that the game can not be completed.

### 2.1 Narrative

I want to go as far as to say that narration defines what being humans means. We, as a society and as a species, have always used a storytelling context to receive and deliver information. Be that a story on the dangers of bears and lions to a more modern setting for the sake of leisure. But even then, for the sake of argument we need a proper definition of a narrative. Riedl and Young stated that a narrative is in it's simplest form a temporally ordered sequence of events [12]. This is the definition I will keep to throughout this document. One other thing of particular note; a narrative does not explicitly mean the usage of text or spoken word. This is an easy, and clear way to convey a story,

---

<sup>1</sup>twice development within the space of 3 words. NOT DONE

<sup>2</sup>Does this sound preachy?

but never the only ways to do so. Granted; it can certainly add to the experience, but in my own opinion a game should, first and foremost, try to convey a story through it's mechanics. To discard that rule is to introduce the expensively named *ludo-narrative dissonance*, or a discord between narrative and gameplay. There has been a vast discussion on the Internet about this phenomenon, on which I will not elaborate, but is worth of note to any aspiring game developer/designer. Games can be a powerful medium in which to explore the human condition, but for that we can really come to that point we need to start treating it with that same respect. And that's my preachy part.. which probably needs to be cut<sup>3</sup>.

## 2.2 Static and dynamic generation

Throughout this document I reference to static and dynamic generation, or static and dynamic narratives. These terms refer to the point in time where the generation takes place. When we speak of a static generation, this happens during the development. A developer may choose to generate a game world and continue to populate that world with the rest of his content. In a dynamic setting, said world is generated at runtime, usually when a new game is started. This means that every time the player starts a new game he gets a new world to explore. In a narrative context that means that a static narrative has been generated, but will never change. Whereas a dynamic narrative will always try to be different from the former.

---

<sup>3</sup>Which makes me sad

# Chapter 3

## The interactive story

Most of the research done on narrative generation is contained within the field of Interactive Storytelling (IS). The layman may not see the difference between a game with a story and an interactive story, and can be forgiven for this is still in open debate within their respective fields. However open the discussion may be there are, albeit subtle, differences. This chapter covers the work done in this field that relates to my own research, and discusses the contrast between the conception of interactive storytelling and video games.

### 3.1 Video Games and Interactive Stories

The notion that an interactive story is a distinctly different product than video games has been in open discussion ever since video games started having complex stories themselves. In the early years of video gaming (games like **Pong**, **Pac-Man** and **Tetris**) games usually didn't have a story<sup>1</sup>, but then games like **The Legend of Zelda** by *Nintendo R&D4* became popular that did have a, albeit small, story. At this point in time the distinction, and especially the semantics, became open for debate.

In this day and age contemporary games often don't get shipped without an attempt at a story, with the exception being *social* games like **Candy Crush** by *King* and **Bejeweled** by *PopCap Games*, the lines between an interactive narrative and video games become blurry. As matter of fact; the term *video games* has recently been under fire. The questions; what makes something a game? What defines a game? The web show *Extra Credits*<sup>2</sup> has a nice answer to this:

[...] We're asked this question all the time, but, you know; I think it's the wrong question. It's a distraction. It does nothing but limits us. It's as if we started to ask: "Well is this really poetry?" when poets moved away from rigid meter or rhyming couplets. [...]

The fact is, we are in the middle of a semantic war. My opinion is that in another ten years the industry will have a new term to call these works of interactive engagement. As we stand now games put more emphasis on doing and acting within a set of rules; the game mechanics. Whereas interactive stories revolve around the activity of acting out a story without the constriction of mechanics, usually because the only controls a user has are movement controls supplemented with a contextual button to interact with the environment.

### 3.2 IS

The early attempts to understand interactive storytelling came in the form of **Tale-Spin**[7]. Tale-Spin generated textual stories from data that a user created like; scenery, characters, and the

---

<sup>1</sup>That didn't stop people super imposing stories. See: <http://www.smbc-comics.com/?id=2736>

<sup>2</sup>See <http://extra-credits.net>

problems that needed to be solved. Other work contains the *Oz Projects*[5] that used intelligent agent technology to tackle the challenges in interactive storytelling. In 2006 came the award winning **Faade** by *Michael Mateas and Andrew Stern*[6]. This interactive story focusses on the player who is a close friend of two AI characters. During a cocktail party at the AI characters' home the player can support the characters or try to drive a wedge between them. We can't mention interactive storytelling without mentioning former game designer Chris Crawford, who left the world of video games to work on IS. He wrote *Chris Crawford on Interactive Storytelling*[2] which is a deconstruction of the entire field and compares that with traditional video games. This work is a must read for anyone considering in attempting anything with in the field of IS.

### 3.3 Dynamic plot generation

One of the most cited works when dealing with generating plots for interactive systems. Sgouros proposes a system that generates, manages, and resolves interactive plots[13]. The system dynamically moves the plot forward with the relations and interactions between actors serving as input. Figure 3.1 shows us the flow of Sgouros' *Plot Manager*; the central piece of his interactive story system. Sgouros shows us how we can abstract concepts like relations and events both causal and temporal and defines a syntax used to describe actions actors can make and goals they might have. His system generates so-called Aristotclian plots, where a conflict between antagonistic forces develops out of the initial situation. The plot will move through a sequence of conflicts and always terminates in a unambiguous solution.

### 3.4 Character-based storytelling

At the turn of the century we saw a shift from general story generation towards a more character driven approach. Researchers like Riedl and Mark[11][9] and Marc Cavazza[1] wanted to make characters within the story context be more outspoken and distinct and worked towards systems that made this possible. Both utilising the Mimesis system to generate character-centric plots.

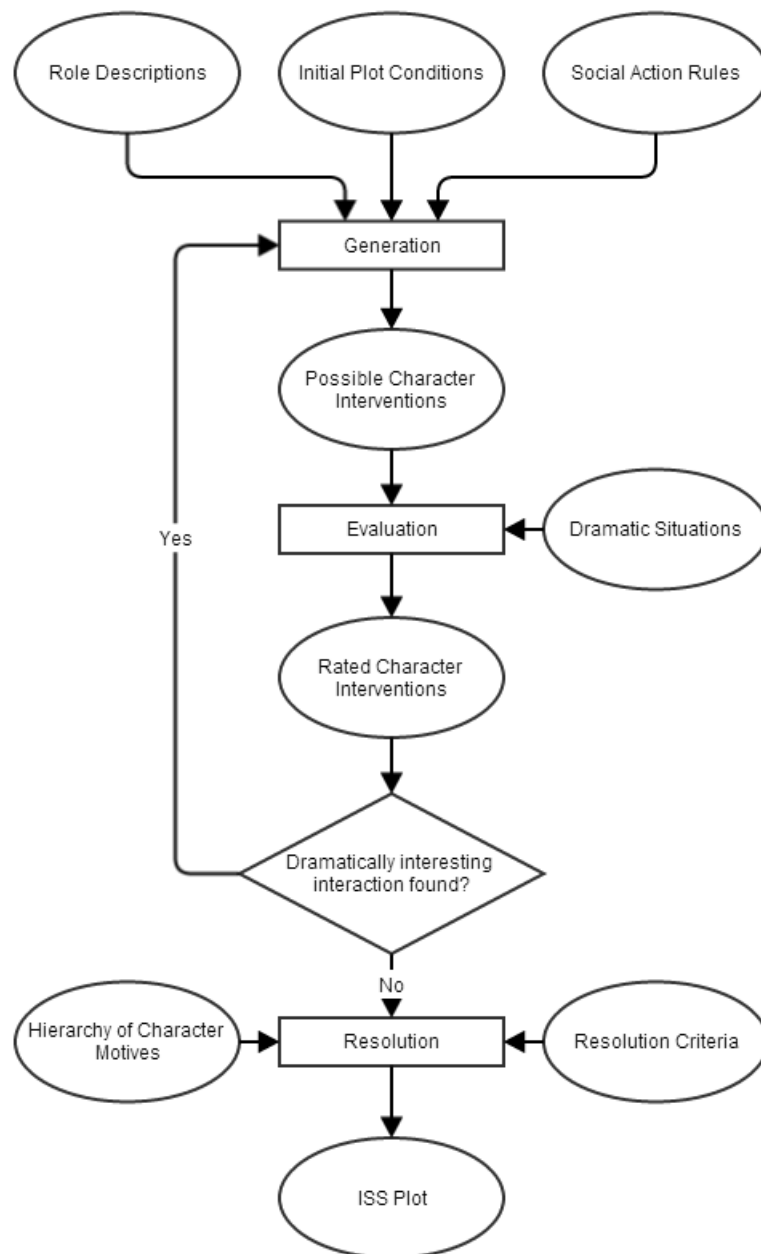
### 3.5 Riedl and Young

Most of my work has been based on and influenced by Dr. Mark O. Riedl and Dr. R. Michael Young who contributed a lot of work to the computational planning of a story. With their papers[11][10][12] they tackle a multitude of challenges from story planners that use character intent to dealing with interactions between users and agents. They co-authored the IS system *Mimesis*[15], that allows storytellers to use a author-centric approach to generating different stories. My own planning system (further discussed in chapter 4) has taken a similar route as the systems proposed by Riedl and Young.

### 3.6 Generative narrative within games

I stated at the beginning of this chapter that most research on subject of narrative generation takes place within the field of interactive storytelling, but there are some commercial games that have used a form of generative narration. In **The Elder Scrolls V: Skyrim** by *Bethesda Game Studios* the developers implemented a system they called *Radiant A.I.* that tries to dynamically react to the player's actions. For example; players with a high *Pickpocket* skill would get told to "keep their hands to themselves" by certain guards. The *Radiant Quest* system expands on this, giving players quests that suit their skills and send them to places previously unvisited. This context is usually used within games; representing a 'fixed' story, but with subtle variance that lets each player have their own experience within a game world.



Figure 3.1: Flowchart of Sgouros' *Plot Manager*

## 3.7 Conclusions

The related work done within this field is vast. So vast perhaps, that to dissect it all greatly exceeds the scope wherein this thesis operates. But most of the research has been focussed on generating a narrative that is "fixed" from the beginning with some leeway towards actual context. The usage of a narrative planner is almost universal to ensure any kind of plot coherence, as is a form of author input to control the flow of a story. These basic elements create a interactive story system that can deliver powerful narratives with believable characters. My own concern is the involvement of the player. In most examples the player is merely an actor with a more volatile behaviour, instead of his own character. I strive for a system that really revolves around the player. The story made by the actions that the player did or did not do.

# Chapter 4

## Narrative planning

A lot of related work has gone into the use of a planning system that decides on the narrative structure. Researchers like Riedl and Young [11, 10, 12] and Cavazza [1] have been researching the use of artificial intelligence for years, and have made some interesting planning systems such as *Memesis* [15]. To help me answer my questions, I have developed my own narrative planner that tracks and generates plot points that the player can resolve to further the character's and the world's story.

### 4.1 Actor attributes

The most influential system within *Diage* are the actor attributes that specify what actors can do and how they should behave. On first glance these attributes are simple key-value pairs that can be used to describe a myriad of character statistics. Looking further the attributes can also describe behaviours that are executed whenever an actor interacts with another actor. In essence the actors don't really interact with each other, but rather their attributes do that for them. A simple example of an attribute would be a *strength* statistic that describes how much damage the actor can do to another entity (see figure 4.1). Another example can be to add personality archetypes to an actor. Looking at the *Dungeons and Dragons* alignment table in figure 4.2 we could specify that the *Lawful Good* archetype will be friendly to all other actors on the *Good* and *Lawful* axes, but will become increasingly antagonistic towards an actor that veers towards the *Evil* and *Chaotic* side.

Besides specifying the actions and behaviours, the attributes can define actions that the narrator can use when planning a new story step. By looking at all attributes within the given setting the narrator can assign actors to help or obstruct the player in his current mission.

### 4.2 Initial Generation

Depending on the input given and the desires of the developer, *Diage* does the initial generation of the game world and plots in varying degrees, as seen in algorithm 1. If no input is given at all, *Diage* will just generate a random amount of entities to populate the world. Input can be given either within the game code, or with a DML file (further discussed in the next chapter). Whether specified or not, the player character needs an initial constraint. This constraint is his/her first 'objective'. This initial constraint can be given within the input, otherwise *Diage* will take an entity within the world and sets that as player constraint. Within the algorithm we see a section dedicated to custom rules. These rules can manipulate anything within the story setting. As an example I created a rule that randomly sets entities to a space as seen in procedure *PopulateSpaces*. The given example is purely non-deterministic, but is a simple matter to populate the spaces more evenly with the entities. This rule system is used throughout *Diage* as a generalized form to give the developer more control on what a given entity can do.

DiageAttribute
+ valueChanged: AttributeEvent + handleAttribute: AttributeEvent + plannerRules: PlannerRule[] + name: String + value: float + maxValue: float + modifier: float
+ HandleAttribute(DiageEventArgs e): void + AddRule(PlannerRule)

Figure 4.1: Class description of the DiageAttribute

GOOD				
LAWFUL	<b>Righteous</b> You strive to create a world order where the weak are protected.	<b>Philanthropist</b> You act within the laws, customs and structure of society to benefit others.	<b>Benevolent</b> You believe in doing good for the sake of good.	<b>Vigilante</b> Right must triumph over evil, by any and all means.
	<b>Dedicated</b> You follow the code of a cause that you believe to be just.	<b>Law Abiding</b>	<b>Kindly</b>	<b>Free Spirit</b> You do as you please, but try not to harm others – or seek to atone for it afterwards.
	<b>Obedient</b> You believe orders should be followed and trust the motives of your superiors.	<b>Selfish</b>	<b>Aggressive</b>	<b>Violent</b> You do as you please, with little concern for how your behaviour affects others.
	<b>Fascist</b> You believe in a world order that puts you at an advantage over others.	<b>Domineering</b> You believe that others should do as you say.	<b>Deceitful</b> You seek to further your own cause by manipulative and underhand means.	<b>Destructive</b> You have no concern for the rights, safety or moral code of others – merely your own superiority.
EVIL				CHAOTIC

Figure 4.2: *Dungeons & Dragons* alignment sheet

```

Input: entities; customRules;
Output: Initial world state
let entities be all entities within current world state;
let customRules be the custom behaviour as specified by the developer;
foreach rule in customRules do
  | rule.Invoke();
end
if entities.count  $\leq 0$  then
  | entities = GenerateRandomEntities(max);
end
if player.constraint == null then
  | select random e from entities;
  | player.constraint = e;
end

```

Algorithm 1: Initial planning

```

/* Populates the spaces with the current entities within world state */
Input: entities; spaces;
Output: All entities are randomly moved to a space
let entities be all objects within current world state, excluding spaces;
let spaces be all spaces within current world state;
while entities.count  $> 0$  do
  | select random s from spaces;
  | entities.pop().MoveToSpace(s);
end

```

Procedure PopulateSpaces

### 4.3 Step Generation

When the initial generation is complete, the player should carry out his/her objective. When the objective has been completed, the planner will automatically start a step generation<sup>1</sup>. This generation is like the initial generation but uses the actor attribute system and the player as extra input. The attribute system will be covered in a later section. This step generation looks at all the given variables and generates a new constraint for the player(see procedure GenerateConstraint). The longer the character's life, the more exact this generation will be. For any action taken by the player can be used in the generation. It must be said that the actions that the planner can take needs to be specified by the author.

```

Input: entities; actors; player; customRules;
Output: new world state
let entities be all entities, excluding actors and the player;
let actors be all actors, including the player;
foreach actor in actors do
    | actor.rules.Invoke();
end
/* It's possible that one of the rules gave the player a constraint, so we'll
   check */
if player.constraint == null then
    | GenerateConstraint(player);
end
foreach rule in customRules do
    | rule.Invoke();
end

```

**Algorithm 2:** Step planning

```

Input: player
Output: new constraint for the player
if rules.count  $\geq 0$  then
    | foreach item in rules do
        | rules.invoke(player);
    | end
else
    | select random e from entities;
    | player.constraint = e;
end

```

**Procedure** GenerateConstraint(Player)

### 4.4 Events

Riedl and Young[12] and Julie Porteous and Marc Cavazza[8] have demonstrated that stories are perfectly suited to be represented as a sequence of temporal and causal events. Porteous and Cavazza suggest to use events as a constraint for partial temporal order. Defining operators as the following:

---

<sup>1</sup>in want of a better word. The term *step* comes from cellular automata, which I'm currently working on

operator	meaning
<i>sometime-before a b</i>	<i>b</i> must be made true for the first time before <i>a</i>
<i>sometime a</i>	predicate <i>a</i> must be true at some stage of the narrative
<i>at-end a</i>	predicate <i>a</i> must be true at the end of the narrative

These operators ensure that we have some control and gives us a partial temporal order, partial because not all of the events are ordered with respect to each other. The table is further expanded upon in their paper *Controlling Narrative Generation with Planning Trajectories: the Role of Constraints* [8].

The *Diage* planner uses event constraints like this to layer plots and direct the player on what to do next. In the context of a *roguelike* we don't know when the story is going to stop, as it stops with the death of the player character. That could be within 2 minutes, but it could be several weeks. *Diage* takes this into account by keep adding segments onto a story, sometimes using information gained from previous plots and at other times creating entire new spaces and NPCs therein. So the operators that Porteous uses we use within these small story steps instead of the entire narrative. *Diage* tells us the story of the player character and how he/she influenced the world around him.

## 4.5 Planning flow

The previous sections described the processes that *Diage* goes through when generating a new world. The flowchart in figure 4.3 displays the steps taken by the planner. The initial generation influences the world by setting up spaces, actors, and objects and manipulating these entities by their specific rules. After this, the player can, and will, influence the world by acting in it. The planner keeps track of his/her actions and saves this for future story generation<sup>2</sup>. The step generation gets activated automatically when the player's constraint is removed, i.e. a plot is resolved. This can also be manually activated by the author on whatever occasion or event he/she wishes. After the step generation the player gets his/her new constraint and, depending on the generative rules used, the world can be manipulated too. Be that actors that move, or new spaces that are generated. The step generation is highly dependant upon the player's attributes, as everything can react to the gain or loss between story steps<sup>3</sup>. The actor attributes are a powerful tool in the *Diage* arsenal, and the (game)world literary revolves around them. It has to be noted, that the design of *Diage* works with semi-persistent worlds too. The world data can be saved to be used in a later play-through, giving the world a whole epic of one player, and starting the next.

## 4.6 Narrative planning in Rouge

The Narrator plays an important part within Rouge. From initial world building, to near continuous step generation of the story. This step-generation takes place every turn, ensuring fast iteration on the world information present and quick building of character and world history. The character specific history is used to construct behavioural patters, and remembrance of past interactions with the player character. The latter is most interesting when using persistent worlds, where a new player character will hear stories of a former character's story. The goal is to eventually create legends and myths surrounding previous characters and their influence upon the world. The world history is to create an awareness of the world that was. Actors can reference locations that were previously infested with monsters, or a city that was destroyed.

<sup>2</sup>For example; a foe thought defeated returns for a rematch.

<sup>3</sup>Mental note: one of the papers referenced this as a beat, might be a good idea!

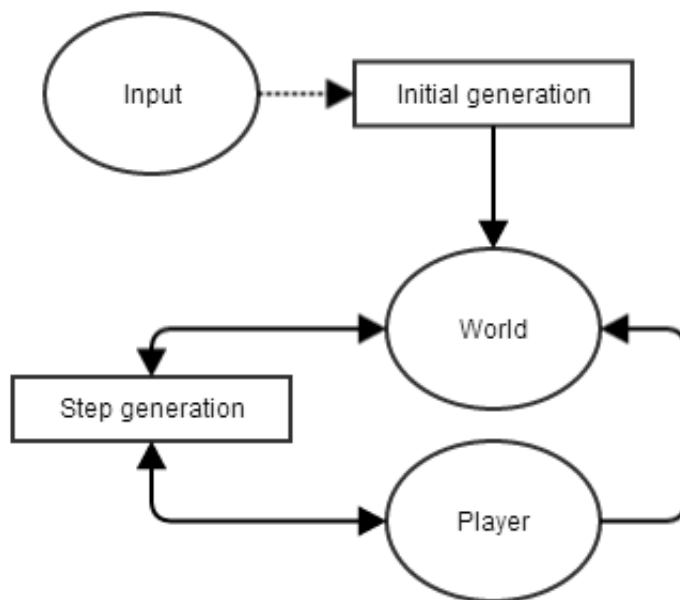


Figure 4.3: Diage Planner flowchart



# Chapter 5

## Diage Modelling Language

In this chapter I will cover the Diage Modelling Language (DML) that is used to visualize the flow of information, some are static and others will wait for the interaction of the player to release this information and ensuring plot progression. *Diage* uses the symbols to represent the *Diage* entities as seen in figure 5.1.

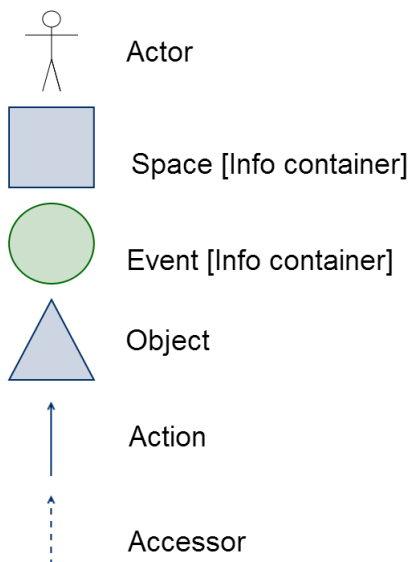


Figure 5.1: DML Symbols

### 5.1 Entities

*Diage* entities come in four forms; **Actors**, **Objects**, **Spaces** and **Events**. The latter two are also information containers which I will discuss in section 5.2. The following section will only cover the pure entities; the objects and actors. In conjunction with **actions** and **accessors** these entities convey story information and plot progression.

#### 5.1.1 Objects

*Diage* objects form the basis of all entities. They represent the props and items that we find in the world that have narrative importance. For example; if the Player walks in to a shop *Diage* does not specify all items that one could buy in the shop, but only those that have plot importance. In terms, these objects should adhere to the *Checkhov's Gun* principle. This dramatic principle states that all objects used in a narrative should eventually be used. I quote: " *One must never place a loaded rifle*

on the stage if it isn't going to go off. It's wrong to make promises you don't mean to keep.<sup>1</sup> Objects have three properties; a **name**, a **ID** and a **type**. The **ID** is a unique identifier dependant on the type. And the type is used with actions/accessors as seen in figure 5.2 (Further discussed in section 5.1.2). The name is the noun given to an object within the story context. For example; The player receives the *Skeleton Key of Awesomeness*, but the type is just **key** giving it no special properties than any other key. It might make sense to name an object something else than it's type, but a name is not given it defaults to it's type. Objects form the world, and all other entities derive from the *Diage* object. This means that all entities have the same properties as the object, but can extend upon it.

### 5.1.2 Actors

An actor is the representation of any one object that can, as the noun implies, act. Examples are the store-clerk, a wandering adventurer or the player. The actor is the only entity that can physically interact with the world, and by doing so the only that can change the world's state. By being able to change the world, the actors are the only entities that can ensure plot progression. Just like the object, an actor has three properties; a **name**, a **ID** and a **type**. The type property is used in predefined actions as seen in figure 5.2a. This figure defines that the **Player** can **speak** to all actors of type **NPC**. A further glance at figure 5.2 shows some more actions that could be defined for the player actor. These predefined actions tell us that the player can trigger all events and enter all spaces. I will expand upon these actions in section 5.3.

## 5.2 Information Containers

Information containers are entities that hold story information. A space holds information about it's spacial children, and events release information into spaces when they are resolved. Information containers have the unique property that they are nestable. For example; a space that represents a city can hold several spaces that represents housing.

### 5.2.1 Spaces

A space is the representation of any segment of the world or the world itself. As spaces are info containers they are nestable, as mentioned before, but they differ in the fact that they can hold every entity as a child. These children make up the spacial awareness of the space and tells us what information it can pas on to actors. Usually an actor gains all the information a space can give upon the moment it enters the space; when the actor becomes a child object to the space. This can be modified, and some parts of information maybe withheld from the actor, but this is where the events come in.

### 5.2.2 Events

A event is the odd one out as an entity, as it is the only one that does not represent something within the narrative. A event is the abstraction of information that is released into the story when an actor - usually the player - interacts with it, thus events are used for story pacing and narrative convenience. When we need the state of a space to change we use a event to initiate that change. This only applies on the narrative context of the world, because *Diage* does not specify everything that happens with in the interactive context. For example; if the player went to a store to buy some cheese to eat. *Diage* only specifies the store's loss of the cheese, if said food item is a special narrative item. Like a poisonous piece of cheese that the villain left there as a cunning trap for our hero. Events make the world go round and are the dynamic forces in *Diage*.

---

<sup>1</sup>[http://berlin.wolf.ox.ac.uk/lists/quotations/quotations\\_by\\_ib.html](http://berlin.wolf.ox.ac.uk/lists/quotations/quotations_by_ib.html)

## 5.3 Actions and Accessors

Actions and accessors are the abstract connectors in *Diage*. They convey what actors can do (actions) and what knowledge they possess (accessors). Some accessors are implied, due to the fact that an actor might be the child of a space, in other cases these connections are explicitly added to a *Diage* model. If we review the diagram in figure 5.3 we see that the mayor has no connections whatsoever. This would imply that the Mayor has no knowledge about what's or who's in the store. If we compare that figure to figure 5.5, we see that the Mayor now has a connection to the Shop, thus we can be sure that the Mayor has the knowledge it would have, if it had been in the shop space itself.

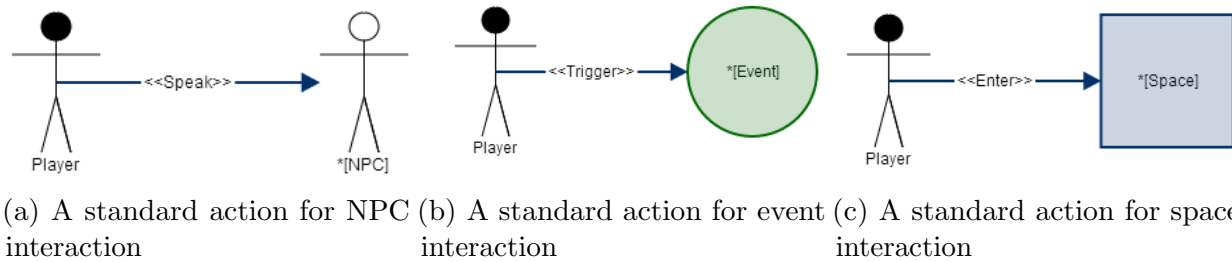


Figure 5.2: Predefined actions

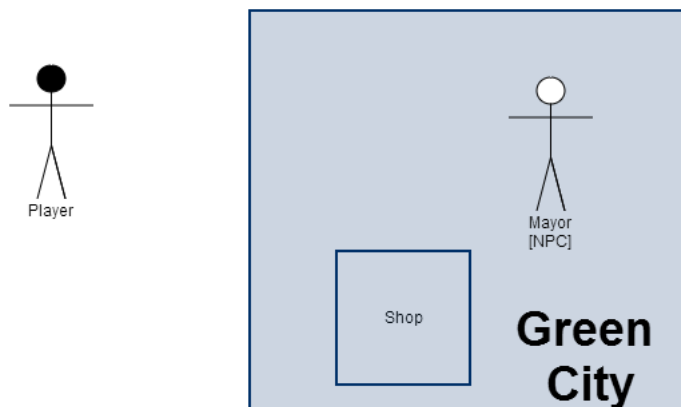


Figure 5.3: An example of a Diage diagram using predefined actions

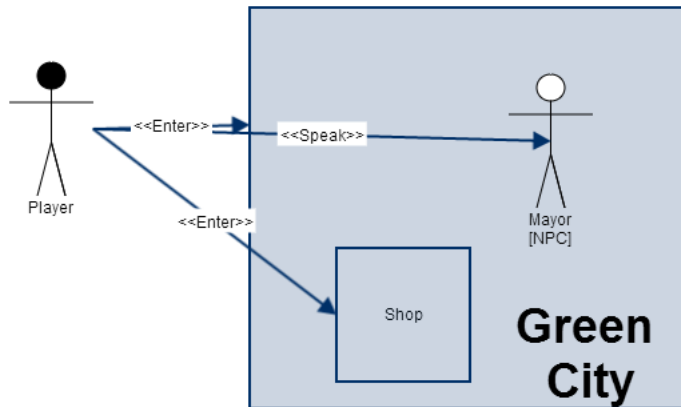


Figure 5.4: An example of a Diage diagram without using predefined actions

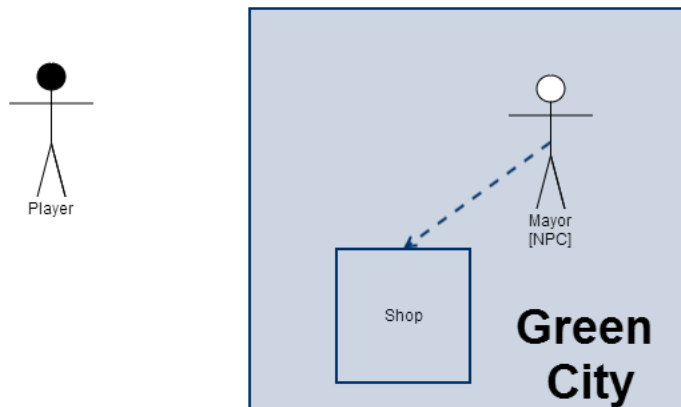


Figure 5.5: A *Diage* diagram showing a explicit connection between the Mayor and the Shop

## 5.4 DML Creator

For the use of DML as input for the Narrator, it had to be convertible to a format that is computer readable. While any graphing software can be used to create a DML diagram, I created the *DML Creator* for ease of use. The DML Creator is used to create and export/import DML diagrams as *JSON* files. The DML Creator can also specify the attributes an entity has. **\*\*SCREENSHOTS!!\*\***

# Chapter 6

## Rouge

*You are a nameless traveller that has been compelled to search for something. An object, a person, or just a place. You don't know yet, but it's up to you to find out. You will travel through various caves, forests and towns to pursue the whims of your heart. Every time you seem to solve a problem, another one turns up. Always moving you towards some inevitable doom. Are you born for heroism, or will you die unloved and unknown?*

*Choose your path, and see where the voices in your head take you.*

Rouge is a *roguelike* tile-based game that is created specifically for the demonstration of the *Diage* narrative generation system. Created within my own framework *SilicaLib* created on top of the XNA game-framework created by Microsoft. Rouge is characterised by the fact that the world and the narrative is generated by the direct influence of the player. This section elaborates on how Rouge was made, what all the different algorithms are that were included, and gives definitive answers to my research questions from Chapter 1.

### 6.1 Mechanics

The game mechanics in Rouge are fully implemented within the *Diage* attribute system. When ever actors interact with the world their attributes can effect the world according to their rules. Whenever they interact with other actors, these attributes will influence dialogue or cause either an antagonistic, friendly or neutral response in accordance to their moral alignment.

Actors move around the world in turns. When an entity gets the turn, its actions points are restored and he can use these to either move around the game world or attack an enemy. Attacking an other actor results in a damage step. This step subtracts the strength attribute of the attacker from the defence attribute and subtracts that outcome from the defenders health attribute.

Normal interaction with other actors does not cost an action point and resolves in a simple conversation. The player asks the other actor for any information regarding his current quest. If the actor knows about it, it will give either helpful or hurtful information, depending upon his alignment attribute. The alignment resolution can also result in the actor refusing to answering the player's question if their alignments are to far apart from each other.

### 6.2 Cellular Automata

The world within Rouge is procedurally generated by a cellular automaton. Celular automata work by defining a regular grid of cells, each in any number of states. Using rules pertaining to the neighbours of each cell, that cell can change its state. For example; we define the states *alive* and *dead*. After that we define several rules:

1. A cell goes into a *dead* state when it has less than 2 live neighbours.
2. A cell goes into a *dead* state when it has more than 3 live neighbours.
3. A cell goes into a *alive* state when it has exactly 3 neighbours.

With these definitions set the grid could be populated with a random amount of cells in live state, assuming that the dead state is the default. Each step of the automaton will cycle through the rules, checking to see if the conditions of a rule is set. After all the cells have been checked the rules execute and changes the grid. The truly awesome thing that cellular automata bring with them, is the potential to emergent behaviour. With these three simple rules, we can generate moving shapes, and repeating patterns. The rules come from one of the most recognized cellular automata; **Conway's Game of Life**. Figure 6.1 shows an example of the emergent behaviour that these simple rules generate.

During a step of this automaton, the algorithm walks through the game world and checks each tile for certain conditions pertaining to their neighbours. If these conditions are met, his state will be changed accordingly. The states a tile has in Rouge are a simple *alive* or *dead* state. When a tile has 5 or more live neighbours, the tile becomes alive themselves. However, when a tile has less than 2 live neighbours, the tile dies. The first rule ensures that tiles group together, and the other rule destroys any singular 'island' tiles.

```

Input: tilemap
Output: new tilemap
let deadCells and liveCells be an empty collection of tiles;
foreach tile in tilemap do
  | if tile.neighbours  $\geq 5$  then
  |   | liveCells.push(tile);
  | else if tile.neighbours  $\leq 1$  then
  |   | deadCells.push(tile);
end
foreach tile in deadCells do
  | tile.alive = false;
end
foreach tile in liveCells do
  | tile.alive = true;
end

```

**Algorithm 3:** Cellular Automation algorithm as used in Rouge

## 6.3 Flood filling spaces

As a recap, one of the *Diage* entities is the space. Spaces in *Diage* are representations of any game-world<sup>1</sup> space, be that the entire world, a city, a shop or a singular room. Spaces in Rouge are designated areas where the player can find shops and resolve his/her objectives. These 'rooms' are generated using a simple tile-based flood-fill algorithm that circles round a selected tile. The algorithm has a maximum allowance that gets depleted by a tile's cost. This cost is higher for tiles that have fewer live neighbours than those that are encircled by them. This creates an the effect that corridors are more expensive to fill (thus have a lower grouping rate) and large open spaces are easy to fill all the way through.

---

<sup>1</sup>hyphen?

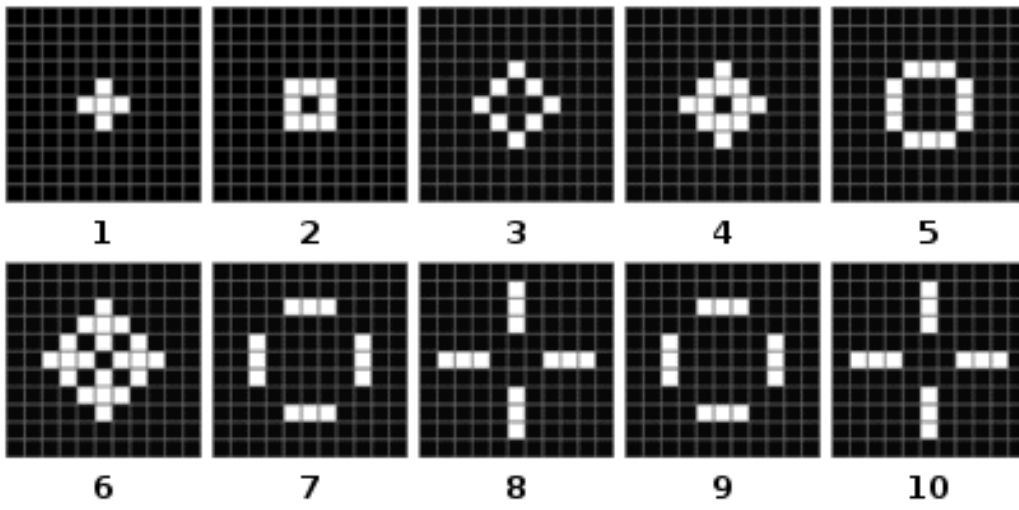


Figure 6.1: An example of the Game of Life generation steps

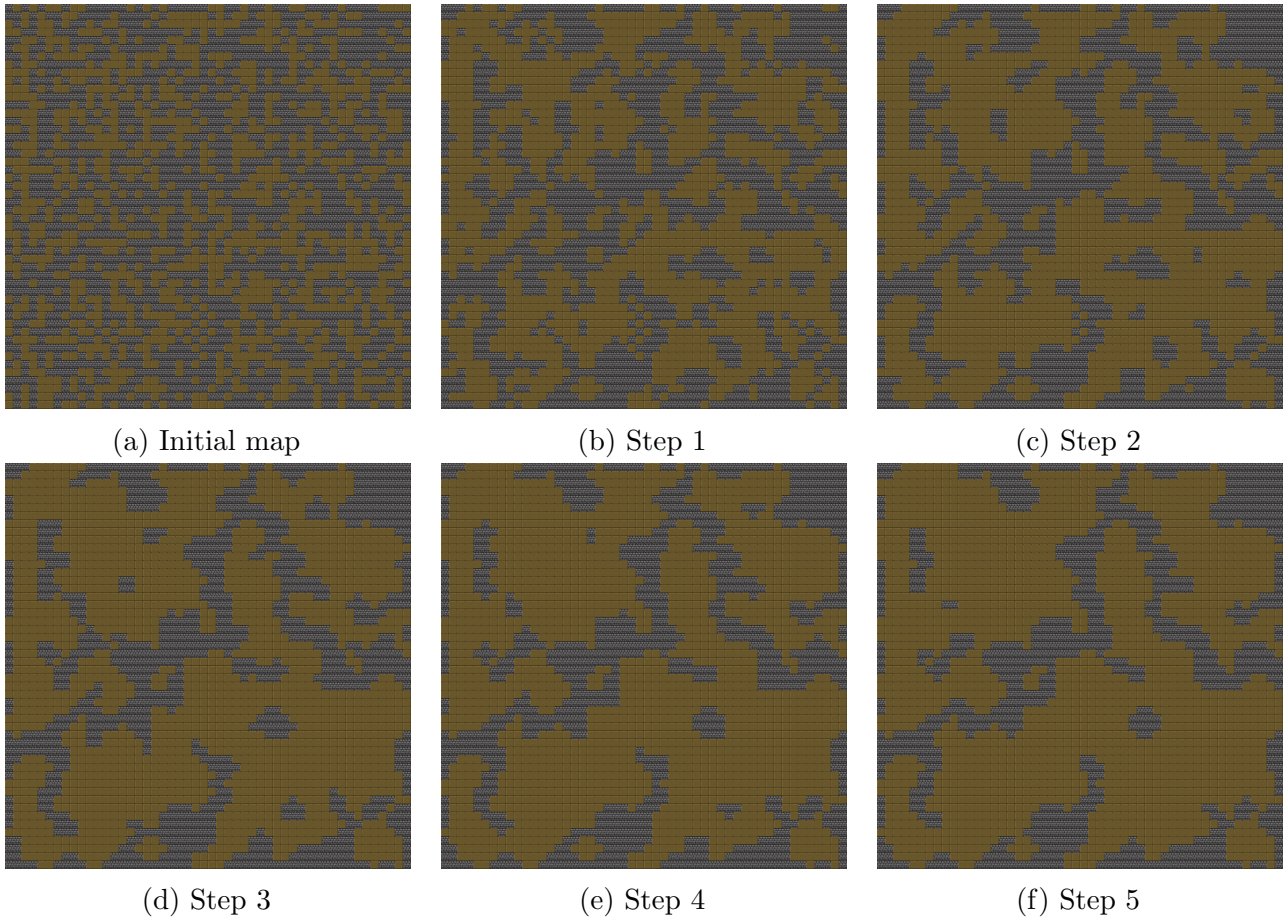


Figure 6.2: The generation of a 50 x 50 tilemap with 20 x 20 (pixels) tiles





# Chapter 7

## Conclusion

In this chapter I conclude my research and give some ideas for future study.

When I started my research, I asked if *roguelike* games can benefit from a procedurally generated narrative. I ask these questions from a designers point of view; "*Does the game get better/more interesting?*", and from a developers perspective: "*Can we develop games faster/more efficiently?*". In the main text I gave multiple answers to these questions and concluded that they both have merit.

I hold that procedural narrative has value within video games. In 20 weeks I've built a system that enables a developer to create games that generate an emergent narrative as a result of simple values given to simple representations. While true that my research focuses on one type of game, the field is open for further work into other genres. *Diage* itself can help developers build *roguelike* games within a smaller time-frame, due to the fact that *Diage* handles most of the story-writing.

Using *Diage* a designer could design a game around the ambiguity of the context, with the inherent fact that every play through can give a player a sense of an actual new adventure. Even without the use of *Diage* as a narrator, the attribute system allows designers to quickly design and test any game that has a implicit use of numbers, such as RPGs, Dungeon Crawlers and Tactics games.

### 7.1 Further Study

As mentioned before, *Diage* was limited to *roguelike* games for the sake of scope. Other video game genres might be able to benefit from the use of a narrative system; especially games that are usually narrative-heavy.

*Diage* is normally expressed with a DML diagram, but the underlying structure is that of a graph. Little work has gone in to using this graph as a means to structure the direction of a narrative. More research in respect to graph translations in relation to *Diage* could result in a better way to direct the narrative into a specific direction. As example; the designer could define an initial state, and an eventual state. Using graph translations *Diage* could interpolate several *in-between* states that the narrative will take. A method could be developed in which *Diage* could output all variants of narrative trees respectively to the choices available to the player. Displaying all possible endings a discreet story arc could have.

It has been pointed out to me that the *Diage* system could be implemented in such a way that it could be used by the *Mechanations* system developed by Dr. Joris Dormans<sup>1</sup> ...

---

<sup>1</sup>A bit too butt-kissy?



# Chapter 8

## Reflection

This section will cover my reflection upon the process of creating this document and all the work that relates to it in the style of a post-mortem.

### 8.1 Diage

Diage stands for *Dialogue Generator*, which was the first idea that I had when starting this project. I wanted to make a generator that would look at the world state and create a dialogue that had impact on the context that the player was in at this moment. After several weeks of reading up on the related work, I switched the project to a more full narrative generative system. This was in part of it seeming more interesting and also somewhat easier to accomplish within the time frame I had. I really like the name *Diage*, so I stuck to it, even if it did lose it's meaning.

### 8.2 The literature

As previously mentioned; I spent weeks on reading literature. I've spent the better part of 10 12 weeks reading up on related work and developing the *Diage Modelling Language* as a frame of reference. While the amount of reading is not unprecedented in a research study, it did bring the project in danger of finishing on time. After I did start working on what would become the Narrator I had misjudged some aspects of the event system which resulted in having to throw away a lot of work in early December, with less than 8 weeks until the deadline.



# Bibliography

- [1] Marc Cavazza, Fred Charles, and Steven J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24, July 2002.
- [2] Chris Crawford. *Chris Crawford on interactive storytelling*. New Riders, 2012.
- [3] Marvin Katilius-Boydston. The semiotics of A.J. Greimas: An introduction. *Lituanian quarterly journal of arts and sciences*, 36(3), 1990.
- [4] Brian Magerko, John E. Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. Ai characters and directors for interactive computer games. In *Proceedings of the 16th conference on Innovative applications of artificial intelligence*, IAAI’04, pages 877–883. AAAI Press, 2004.
- [5] Michael Mateas. An oz-centric review of interactive drama and believable agents. Technical report, 1997.
- [6] Michael Mateas and Andrew Stern. Faade: An experiment in building a fully-realized interactive drama, 2003.
- [7] James R. Meehan. Tale-spin, an interactive program that writes stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98, 1977.
- [8] Julie Porteous and Marc Cavazza. Controlling narrative generation with planning trajectories: The role of constraints. In *Proceedings of the 2nd Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, ICIDS ’09, pages 234–245, Berlin, Heidelberg, 2009. Springer-Verlag.
- [9] Mark O Riedl and R Michael Young. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010.
- [10] Mark Owen Riedl, C. J. Saretto, and R. Michael Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS ’03, pages 741–748, New York, NY, USA, 2003. ACM.
- [11] Mark Owen Riedl and R. Michael Young. Character-focused narrative generation for execution in virtual worlds. In *Proceedings of the International Conference on Virtual Storytelling*, pages 47–56, 2003.
- [12] Mark Owen Riedl and R. Michael Young. An intent-driven planner for multi-agent story generation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS ’04, pages 186–193, Washington, DC, USA, 2004. IEEE Computer Society.
- [13] Nikitas M. Sgouros. Dynamic generation, management and resolution of interactive plots. *Artificial Intelligence*, 107(1):29 – 62, 1999.

- [14] Peter William Weyhrauch. *Guiding interactive drama*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1997. AAI9802566.
- [15] R Michael Young and Mark Riedl. Towards an architecture for intelligent control of narrative in interactive virtual worlds. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 310–312. ACM, 2003.