(WORKING PAPER)

The Meaning Potential of Procedurality:

initial considerations on **procedural** semiotic resources

Daniel Peixoto Ferreira*

JUL 2015

*This text was written in the context of the Playing Beowulf and MissionMaker projects, developed in the DARE Collaborative at the London Knowledge Lab (UCL Institute of Education), under joint supervision of Prof. Andrew Burn and Prof. Alison Gazzard. Daniel Peixoto Ferreira is a Digital Poetics PhD student at the School of Communication and Arts of the University of São Paulo (ECA/USP), currently at the UCL IOE as a CNPq Brasil Fellow. More information at the author's research blog (`7luas.com.br/research`) and at the DARE website (`darecollaborative.net`).

Abstract

This text presents initial considerations about **the meaning potential of procedurality** in digital media, from the perspective of the designer working in this medium. In simple terms, the **procedural aspect** of digital media refers to this medium's fundamental ability to execute a series of rules in an autonomous fashion. **Procedural authorship** allows for unique strategies of representation and expression, different than in any other traditional medium or language. The objective of this particular study is to identify and analyze the usage of **procedural semiotic resources** by the designers of digital games created using MissionMaker, as well as the availability of such resources in that software, as a means to discuss **the meaning potential of procedurality** in a more general sense. The framework used is the Multimodal Analysis approach, as well as a methodology of my own, developed in a previous research.

Keywords: *procedurality, authorship, expression, multimodal analysis*

1 Introduction

This text presents initial considerations about **the meaning potential of procedurality** in digital media. This is done from the perspective of the designer exercising **procedural authorship** in this medium, as opposed to the moment of reception or interaction with the finalized work.

In simple terms, the **procedural aspect** of digital media refers to this medium's fundamental ability to execute a series of rules in an autonomous fashion. This allows for unique strategies of representation and expression, different than in any other traditional medium or language (the term "traditional" is used here in opposition to digital media, to refer to previous mediums and languages). Also, because of its status as a metamedium, digital media allows the materialization of content in a variety of different languages, such as visual, sound, musical, kinetic and so on.

The concept of procedural authorship is used here specifically in reference to the practice of the designer or creator dealing directly with procedural content. This is fundamentally different from the use of procedurality solely as a tool (which is the case of any activity done using a computer, for example), as we will see later.

In this particular study, the focus will be on digital games created in a software called MissionMaker, in the context of the **Playing Beowulf** pilot workshop. The purpose of this workshop was to explore the Anglo-Saxon epic poem Beowulf via a series of creative exercises in a variety of different languages and mediums, including drama, improvisational theater, creative writing, drawing, animation, film and digital games. The analysis of the games involved the observation of the creative process, analysis of the finalized games (and project files) and conversations with their creators.

By analyzing these games, the objective is to identify and describe the usage of **procedural semiotic resources** by the designers using MissionMaker, as well as the availability of such resources in this particular software, as a means to discuss and investigate **the meaning potential of procedurality** in a more general sense.[1]

MissionMaker is a computer game authoring tool for creative learning, currently in development at the London Knowledge Lab (UCL IOE). It allows the creation of games composed of different locations, objects and characters, as well as the definition of rules, behaviors and properties for these elements.

The tool distinguishes itself from most game authoring software because of its particular workflow, which allows even new users to have a simple, but fully playable, environment almost immediately. It is then possible to customize and add new elements gradually, as the program's interface and features are explored by the user. This approach makes MissionMaker particularly suitable for educational purposes.

[1]A more elaborated description for the concept of meaning potential would be: a particular idea represented by a creator in the form of an object or experience, and which has the potential to be realized and interpreted by an audience in a certain context.

The poem **Beowulf** is set in Scandinavia, and it narrates the adventures of the hero Beowulf, as he helps the tribe of the Danes to protect themselves from a mysterious and horrendous creature known as Grendel. Beowulf is able to defeat Grendel, bringing the wrath of the creature's mother, which is also killed by the hero. We will see later on that the poem features not only an exciting plot, but also a rich world, with many engaging settings and characters, as well as a particular narrative style and use of language (the version of the poem used for the workshop was the modern English translation by Frances B. Grummere).

Overview

The following is an outline of this text, describing the main concepts and themes presented in each section (section 1 is this introduction).

Section 2 (p. 6) is an introduction to the concept of Procedurality, and the practice of procedural authorship. This includes a brief presentation of the particular perspectives that other authors have on this subject, the practices and perceptions of the creators and audience in relation to it and some of the most common misconceptions regarding this topic, which this study aims to clarify. I also briefly present the concept of procedurality in a more general sense, before (and outside) digital media.

Section 3 (p. 16) introduces and presents the Framework and Methodology used for the analysis of the MissionMaker games, as well as the software itself. This is based on concepts and theories from the the Multimodal Analysis approach, as well as a methodology of my own.

Section 4 (p. 21) consists of the actual case study analysis, in order to identify and discuss the procedural semiotic resources in MissionMaker. First, the section presents initial elaborations about computational logic, relating it to this particular software - this includes concepts such as levels of abstraction, execution of the code, multiplicity of contributions in the context of digital productions and the notion of different possible perspectives of analysis.

Next comes the analysis of the software, beginning with the presentation of the user interface from the perspective of the designer using MissionMaker. I then identify the procedural semiotic resources available to the designer - which are the Variables and Rules - , as well as those hardcoded in the software - in the game engine -, and how they are composed into a finished product.

Finally, I further discuss how procedural authorship relates to the creative process in previous traditional languages and media, based on the Playing Beowulf workshop.[2]

Section 5 (p. 49), which concludes this text, summarizes the findings derived from the analyses, presents what I hope to have achieved with this study and proposes further developments.

[2]The term "hardcoded" refers to elements which are directly included into the program's main code, which is usually into de "game engine" (or just "engine"). The "engine", in turn, is usually the part of a program that deals with the most general - and fundamental - aspects of the experience, such as describing the physics, gameplay, rendering and representational models. It normally does not contain specific instances of data or assets (such as character models, sounds or text) - instead, the engine describes how the content is structured, accessed, displayed and so on. As we will see further on, in the case of a game creation software, elements which are hardcoded into the engine are usually not available for editing or customizing by the user (the game designer).

Some Considerations

The focus of this study, as mentioned before, is on the perspective of the designer, as opposed to the execution of the code or the reception of the final work by the audience. This doesn't mean that these other aspects are not important, or that they are not considered in this study.

The moment of fruition is considered, but mainly (1) as a means to access the perspective of the creator, the creative process or elements and aspects of the digital production and (2) as imagined or planned models and affordances, in the mind of the creator.

Also, the main objective of this study is not the specific analyses of the MissionMaker games in themselves, but what they reveal about the grammar and resources available in this particular software. This is first and foremost a discussion about how this specific tool allows designers to exercise procedural authorship, and the meaning potential that may be represented or expressed through it.

2 Procedurality

This section consists of an introduction to the concept of procedurality, its fundamental properties as well as some common misconceptions. It also includes a brief presentation of how this subject is approached by other authors in the field, and how they relate to my own research.

In simple terms, procedurality in digital media refers to this medium's fundamental ability to execute a series of rules in an autonomous fashion. According to author Janet Murray, procedurality is "[t]he most important element the new medium adds to our repertoire of representational powers" (MURRAY, 1997:274). This view is shared by many others, such as Alan Kay, Ian Bogost and researchers Michael Mateas and Andrew Stern, who state that "procedural authorship is required to take full advantage of the representational power of the computer as an expressive medium." (MATEAS & STERN, 2005:8)[3]

Game designer and researcher Chris Crawford describes the procedural aspect by comparing it to static data (CRAWFORD, 1987). On one side, **procedurality** is based on algorithms or equations, representing rules, or behaviours. It is dynamic, in the sense that these outputs or behaviours may be different each time the code is executed, according to its internal logic or other factors, such as user interaction. Procedurality is the basis for all the operations executed by a computer, from the high-level functions of software applications all the way down to the operating system and the hardware logic, including dealing with data, outputs and inputs.

On the other side, **static** data consists of pure information, such as numbers, tables, images, sound and text. It is fixed, or linear, in the sense that the content available to the viewer or public is the same every time.

In MissionMaker, as we will see further on, both approaches are available to the game designer. On one side, the software allows procedural authorship in creating the rules and behaviours for the different character and elements of the game. On the other side, it is possible to create and include static data or assets, such as images, sound files and text. The purpose in this study is to focus on the procedural approach.

Strictly speaking, despite the fundamental difference between procedural content and static data, the boundary between them, although meaningful, is essentially arbitrary and not always clear. For example, procedural content can only be materialized (and thus, made accessible to the audience or user) **indirectly**, via content of static nature. So, although the behaviour of the non-player characters (NPC's) in a MissionMaker game is procedural, it is communicated via static means - images and sound. In some types of digital productions, the final product exists **solely** as static content, such as an animation, a sound file or a printed drawing. The latter is the case of the output from the "virtual painter" system Aaron (Harold Cohen, 1973-; Fig. 1, p. 13) - in this kind of situation, since the authorship happens in a metacreative level, via algorithms, the resulting product is still considered

[3]Procedurality as a more general concept precedes digital media, as elaborated further on (p. 14). The central role of procedurality in digital media is discussed in my masters dissertation (FERREIRA, 2011).

to be essentially procedural (this is not the case, however, of most digital games, including those created using MissionMaker).[4]

Another fact to be taken in consideration is that algorithms (or software) are a type of data as well, stored in the computer's memory in the same manner as any other digital file. The difference is that software data, representing **procedural** content, is designed to be interpreted by the computer as a series of instructions to be executed, while **static** data is not.[5]

In fact, every procedural content includes static elements in their composition or structure, in some fundamental level. For example, a rule in a MissionMaker game consists of a simple algorithm which describes how different elements relate among themselves. The elements themselves, however, are usually pointers to static content. Eventually it is possible to have a rule as an argument inside another rule - however, regardless of how many levels of nesting there is, eventually there will be a static element. Thus, the boundary between what is procedural and what is static is not always clear (specially in less structured grammars, such as the case of visual language). Nevertheless, it still holds true that procedurality is fundamentally different from static data.[6]

Chris Crawford suggests the concept of Process Intensity as a measure to represent the proportion of procedurality in a given digital content. This shall be discussed further on.

2.1 Particular Perspectives

As mentioned before, the focus of the research presented here is on understanding procedurality as a language, in a fundamental sense, and from the perspective of the game designer using MissionMaker. This means that the emphasis is on the **algorithms**, or rules, where the designer's intention is represented or codified, as opposed to either the **execution** of the code of the game or the moment of **reception**, or fruition, of the finished product by the player.

Most authors and researchers in the field of digital media share a similar and compatible definition for procedurality. However, each tend to adopt a particular perspective or emphasis on the subject, according to their specific interests or approach. The following is a brief description of the work done by some of these authors, and how they relate to my own research.

Back in 1997, in the seminal book Hamlet on the Holodeck, **Janet Murray** described procedurality as one of the four properties (or affordances) found in digital media, thus distinguishing it from the other aspects - the participatory (referring to interactivity), the spatial (the possibility of virtual navigable environments) and the encyclopedic (the capacity to deal with vast amounts and types

[4]The term "materialized" refers to the form in which the digital production is presented to the audience.

[5]The term "data" is commonly used as a synonym to "static data". The usage of the term "software" also varies. Either it is used in opposition to hardware (meaning any element which is not part of the hardware), or specifically in reference to executable applications (which is how it is used in this text).

[6]What is being presented here refers to the **formal** (or structural) difference between procedural and static data. This means that, in the case of static data, and for the purpose of this classification, the content itself is irrelevant. For example, in an adventure game, even if a particular line of dialogue from an NPC makes reference to a game mechanic, or some other procedural element or aspect of the system, that content (textual or auditory) is still static in a structural sense.

of information; MURRAY, 1997). This was done in the context of the author's research about the development of new forms of interactive drama and narrative. Murray's work is particularly relevant to my research in arguing for the primacy of procedurality, considered by the author to be the property through which digital media can be established as an expressive medium.

Noah Wardrip-Fruin also deals with the expressive potential of code, although with an emphasis on digital literature (WARDRIP-FRUIN, 2006). Also, the author is particularly interested in the moment of execution of the algorithm or program (the "processing"), as opposed to the code itself (the "procedure"). This subtle - but meaningful - difference is reflected in the terminology adopted by Wardrip-Fruin, with the term "Expressive Processing" suggesting that the act of expression (or construction of meaning) happens in the moment of execution of the algorithm. In my own research, the term used is "Procedural Expression" (or “Procedural Expressiveness"), indicating that the expressive quality is represented in the procedure itself.

In **Ian Bogost**'s research, which deals with the rhetorical qualities of videogames, procedurality is described as allowing the modeling of "real-world behaviors into programmatic representations” (BOGOST, 2006:13). This is explained by Bogost through his concept of Unit Operations, defined as arbitrary codifications of particular recurring elements or ideas (in contrast to System Operations, defined as more rigid, totalizing structures), which applies not only to videogames and digital media, but also poetry, cinema and other languages. The author is particularly interested in exploring procedurality as discourse (or rhetoric), borrowing concepts from Literary Theory, Philosophy, Psychology and Semiotics, in order to examine the meanings and implications associated to this approach. Similarly, **Lev Manovich's** cultural studies approach emphasizes the context in which algorithms are created, executed and consumed (MANOVICH, 2001). In that sense, my own research differs from both Bogost's and Manovich's, since I approach procedurality in a more fundamental sense, discussing the expressive potential of the language, as opposed to particular messages and uses.

Artist **Harold Cohen** offers a perspective which is closer to the creator's point of view, by emphasizing the actual computational functions, as well as the autonomy of digital media, which allowed the development of Aaron, the aforementioned “virtual painter" system (COHEN, 2011). Formerly a painter in a traditional sense, Cohen wrote articles explaining the process of codifying a particular art style in the form of algorithms. Another study more closely related to the perspective of procedural authorship is **Michael Mateas**' practice-based research, focusing specifically on the use of artificial intelligence techniques to define character behaviour, as well as narrative structure (MATEAS, 2003a and 2003b). The result of that research is the interactive drama Façade (Michael Mateas and Andrew Stern, 2005; Fig. 2, p. 13), still considered to be one of the most impressive and sophisticated implementations of a procedural narrative.

My own research, as mentioned before, is characterized by a deliberate choice to limit the scope of investigation to focus on the creative process (instead of the reception or fruition of the final product), and on the more fundamental aspects of procedurality (instead of particular manifestations or uses of this approach). Regardless of the differences between these approaches, however, the concept of procedurality is essentially the same in all of them.

2.2 Practices and Perceptions

This study deals specifically with **procedural authorship** in the game design process using Mission-Maker. One of the reasons for analyzing this particular aspect of digital media is that there are still many misconceptions regarding the expressive use of this medium, and how it relates to (and, more importantly, distinguishes itself from) other traditional forms.

Computer scientist Alan Kay once described digital media as a "metamedium" (KAY, 1984). The author argues that the most common use of the computer is as a tool to **emulate** traditional mediums and languages, which means dealing with **static** data, as opposed to **procedural** content. This means activities such as writing, taking pictures, viewing media (such as movies and music) and using the internet to communicate, search for content and transmit messages and files. Although the operations involved in these activities do rely on the procedural aspect of digital media (some in a more intense manner than others), the **content** itself consists of static data.[7]

Whenever the user does deal directly with content that is predominantly procedural, it is usually as a **consumer**, and mostly limited to a few very specific categories, the most evident being digital games, which is the type of productions analyzed in this text. Examples of other kinds of predominantly procedural content include certain digital artworks, such as the interactive piece For All Seasons (Andreas Müller, 2004; Fig. 3, p. 13), editorial pieces, such as Cutthroat Capitalism (WIRED, 2009; Fig. 4, p. 13) and advertisements.

The actual **production** of procedural content, on the other hand, is mostly concentrated among professional software developers, designers and programmers. Interestingly, however, there are particular situations or contexts in which the general user does deal directly with procedural content, although mostly limited to very simple and specific tasks. For example, creating macros in a word processor (to automate repetitive operations) or defining email filters (deciding what actions should be taken to each message according to a series of rules).

However, even in these situations, this doesn't necessarily mean that the users are acknowledging the procedural nature of such content. For example, users who creates macros in word processors don't necessarily think of themselves as being programmers. Even when experiencing procedurally based (or generated) content, certain types of audience will still interpret it via the more familiar grammars of traditional media and languages, such as animation, movies or literature. For example, some videogame players talk about their gameplay experience via the grammar of cinema, describing "scenes" (suggesting a static - that is, fixed or linear - sequence of events), as opposed to behaviours, rules and game mechanics (although this could also be explained by the case of game designers who themselves rely on the expressive strategies of cinema when creating those experiences).

This is why the topic of procedural literacy must be addressed even in the context of computer programming and game design courses. It is not enough to have a student mechanically go through the process of creating environments, characters and rules for a game. It is necessary to make it clear what is being done, and how the process is different than traditional languages and media.

[7]There are surprisingly few statistics available about the usage of computers and digital media. Although there are many studies dealing with specific topics and aspects, and focusing on particular regions and demographics, as of this writing there is no information on a wider scale available.

Of course, there is no "wrong" way to use or enjoy digital media. On the other hand, as mentioned before, many authors argue that the unique expressive potential of this medium, achieved through procedural authorship, is still relatively unexplored, both by creators and the audience (this will be illustrated further when analyzing the MissionMaker games). One of the reasons for that, as I mentioned before, are some misconceptions and simplifications that exist surrounding this subject, which we will see next.

2.3 Some Misconceptions

There are many misconceptions that tend to limit or hinder the understanding of the concept of procedurality, not only among researchers, but also creators working in this field, as well as their audience. This, in turn, may lead to limitations in the development of this approach to its full potential, as well as gaps in the way consumers perceive and interpret such digital productions.

The following are some common misconceptions and simplifications that I identified throughout my research regarding procedurality.

Emphasis on secondary aspects
Certain aspects of digital media tend to be emphasized over procedurality, or associated with the concept, despite not being essential or even exclusive to it. One of the most evident cases of this is **interactivity**, which is an element often found in digital productions, and indispensable in the case of games. However, although interactivity may allow access to procedural systems, it is not a fundamental aspect of this medium.

An example of an artwork that does not include interactive elements, while still featuring an expressive use of procedurality, is the interesting installation Balance From Within (Jacob Tonski, 2012; Fig. 5, p. 13). The work consists of a sofa balanced on one of its feet thanks to an algorithm which, according to the artist, represents "a kinetic metaphor for the inherent risk in social relations" (TONSKI, 2014).

Even in digital games, in which interactivity is fundamental and required - it is the only way through which it is possible to reveal the systems and rules, and to access the content available in a game -, it is still secondary to the procedural aspect. Interactivity not only depends on procedurality to exist, but is defined and programmed through it.

This is why the emphasis of this study is not on the interaction itself, but on the procedural systems (or algorithms) that allow it. For example, in most of the MissionMaker games that were analyzed, Beowulf - the hero of the story - is represented as a strong character, not only through static elements (such as visual cues of his physical strength, or textual descriptions of his greatness), but also through the rules and procedural elements, such as how much damage he can take in combat, how high he can jump and how fast he can move. The meanings being expressed via these rules and behaviours exists by themselves, regardless of the particular situations that emerge in each instance of gameplay.

Considering interactivity as secondary to procedurality in a logical sense does not mean that it is less relevant in an expressive sense. Many researchers, such as Ian Bogost and Lev Manovich, discuss the interactive aspect as a way to approach topics such as ethics, politics and the social role of digital productions (as well as their impact on those areas). However, although these aspects are important and must be considered in any research done in this field, they are not the focus of the study presented in this text. Interactivity is considered here mainly as a means to access procedurality.[8]

Fixed aesthetics

There is also a particular aesthetics that tends to be associated to procedurality, usually though qualities such as abstraction, objectivity and complexity. For example, procedural content tends to be systematical and structured, such as the level design in the videogame Spelunky (Derek Yu, 2008; Fig. 6, p. 13). However, the procedural approach can also produce results which are more organic and fluid, such as the dialogue in the aforementioned interactive drama Façade (Mateas and Stern, 2005).

This is reflected in the system of most of the MissionMaker games analyzed in this study. Usually it is very clear to the player what is the role of each element in the game, and the consequence of every action or situation. Even the more complex and expressive meanings tend to be represented in an objective or systematic manner. For example, a rule may state that if the player attacks another character, it fights back.

However, this is not a limitation of the software's engine itself, but of the designer's approach to the tool. As we will see further on in the game analyses, in some cases the designers were able to create more subjective and organic behaviours.

Restrictions to specific software and techniques

Another common misconception regarding procedural authorship is that this approach is often considered to be restricted to specific software applications, tools or even particular techniques.

The programming languages Processing (Ben Fry and Casey Reas, 2001) and Actionscript (Macromedia Flash), for example, although they tend to be more accessible to inexperienced programmers, can also be limiting. Procedural authorship is an approach that goes beyond particular programming languages or applications, which tend to emphasize certain aspects or uses over others.

As mentioned before, this is a particularly important topic in relation to MissionMaker, which is an educational software for game design and digital media literacy. On one hand, it is productive to focus on one single tool, at least at first. However, it is also important to constantly keep in mind the wider perspective - the fact that most of the concepts used in MissionMaker are also valid in other game creation tools, and even more general purpose software.[9]

[8]Miguel Sicart presents a similar argument for the importance of considering user interaction when analyzing videogames, although in a somewhat radical manner, in his essay titled Against Procedurality (SICART, 2011). While it is essential to take user reception in consideration when analyzing any digital production, specially those that are interactive, the study presented here follows the idea that it can also be useful to focus on procedurality itself, its internal mechanisms and logic.

[9]The same misconception also happens in a smaller scale, with particular techniques, such as the function for generating random numbers, for example, considered by many to be synonymous to procedurality.

This connects with other misconceptions, which are directly related to - or result from - limiting procedural authorship to specific software solutions. Since this study is focused on a specific software, the misconceptions below don't directly apply. However, as mentioned above, by considering these concepts, the game designer using MissionMaker may be able to connect this particular experience to the more general notion of procedural authorship. By doing so, the abilities and theories acquired using this particular software may be further expanded to other software and programming languages, and even to different expressive forms.

For example, procedurality in digital media tends to exist in a limited range of categories, such as generative productions (art and design generated by computer code) or digital games, as well as in certain particular types of content, such as visual, auditory or textual productions. However, there is no practical reason for this limitation, since the applications of this approach are virtually unlimited. For example, consider the striking algorithmic sand sculptures by the artist Jean-Pierre Hérbert (Fig. 7, p. 13).

In procedural authorship, the actual execution of the code also tends to be restricted to certain particular stages of the creative process. Usually, either it is present as the final product itself (which is the case of digital games, for example), or in the moment of generating the final product, which is itself static in nature (such as most types of generative art). However, execution of code can happen in many other stages of the creative process. For example a custom-made software was used for the creation of the choreography of the performance Far (Random Dance, 2011; Fig. 8, p. 13) - in this case, the execution of the code is neither part of the final product nor directly generates it.[10]

Finally, there are particular aspects or elements of a production in which the procedural approach is most commonly found, while others tend to be neglected. For example, the movement of characters in virtual environments is usually mostly defined by static, or predefined, animations, such as a loop for walking, an animation for picking up an object and so on. The procedural approach tends to be used only in specific situations, such as when a character suffers some kind of impact (for example, being hit by a car), in which case a "ragdoll" physics engine is activated.[11]

[10]The different stages of the creative process is discussed further on in the context of the Framework and Methodology section (p. 19).

[11]Although the procedural approach has been increasingly used to define character's behaviour in videogames and other simulations, movements still consist of mostly predefined animations.

Figure 1: Theo (Aaron,1992)

Figure 2: Façade (Mateas and Stern, 2005)

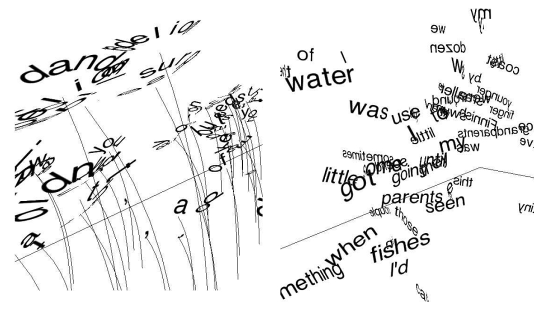

Figure 3: For All Seasons (Andreas Müller, 2004)

Figure 4: Cutthroat Capitalism (WIRED, 2009)

Figure 5: Balance From Within (Jacob Tonski, 2012)

Figure 6: Spelunky (Derek Yu, 2008)

Figure 7: Sand Installation (Jean-Pierre Hébert, 2011)

Figure 8: Far (Random Dance, 2011)

2.4 Procedurality Before Digital Media

The concept of procedurality in a more general sense precedes digital media, and it can be found in (and related to) many traditional practices, languages and media. Children games or play, for example, as well as sports, are essentially procedural activities, in the sense that they involve behaviours and rules.

In the field of poetic expression, Conceptual Art also tends to be procedural in nature, specially in the type of work known as "statements". These are essentially descriptions of artworks, such as One Hole in the Ground Approximately 1' x 1' x 1' / One Gallon Water Based White Paint Poured into this Hole (Lawrence Weiner, 1968) - note how this description (or "statement") is similar to an algorithm (ALBERRO & STIMSON, 1999). The procedural approach is also present in improvisational practices, such as in certain types of comedy, musical performance or theater. For example, in the 1960s, Augusto Boal's Theater of the Oppressed featured improvised scenes, procedural systems that the audience could interact with and explore, as a way to deal with local political and social issues.[12]

Static textual experiences with basic non-linear structures may also suggest a procedural structure, such as the combinatorial artwork Cent mille milliards de poèmes (Raymond Queneau, 1961) or the Choose Your Own Adventure book series. The same can be said of traditional productions that suggest some type of systemic or algorithmic logic. This is the case of the movie The Terminal (Steven Spielberg, 2004), which Ian Bogost argues to be "a unit-operational film about themes of waiting" (BOGOST, 2006:19).[13]

Another example is the aforementioned poem Beowulf, which features rich descriptions of its story world and characters. As mentioned previously, the digital games analyzed in this study were created based on this epic poem.

What makes procedurality different in digital media is the fact that this medium introduces elements such as autonomy, greater level of control, higher structural complexity and more powerful processing capabilities - a quantitative increase that leads to a qualitative leap. The combination of these properties allows for unique representational and expressive strategies, which are distinct from the ones found in any other medium or language.[14]

Having said that, there certainly is much in common between the digital and traditional manifestations of procedurality. For example, they both share a metacreative quality, a fundamental grammar and certain strategies of reception. Therefore, the process of understanding procedurality

[12]Going even further, it is also possible to argue that the creative process itself is procedural in nature, since it happens via thought processes, which are themselves essentially procedural. I elaborate on that topic in my masters dissertation (FERREIRA, 2011).

[13]The Terminal is about a man that, because of diplomatic issues, is forced to live in an airport for several months.

[14]In theory, a digital production such as the videogame Super Mario Bros. (Nintendo, 1985) **could** be translated in the form of a book, in the style of the Choose Your Own Adventure series, for example. The result would be a lengthy book, with each page containing options for every possible action from the player/reader, as well as all the possible changes in the game world (including every NPC and variable). However, the difference between reading that book and playing the original game wouldn't be just the pace of the experience, but its very nature. Such an experience probably couldn't even be originally designed if not in a digital medium.

in digital media as a language in itself can certainly benefit from analyzing and identifying parallels and similarities with traditional practices.

This is reflected in the research presented here, based on the Playing Beowulf project, via the intersection between the practices of videogame authoring (using the MissionMaker software) and creative exercises in other media and languages, such as the dramatic improvisation based on the epic poem Beowulf (which is itself a relatively open text, structured in a semi-systemic manner). This topic will be elaborated further on in the game analyses (p. 21).

The procedural aspect of digital media affects and influences modern life in many ways. However, to many people this is still not a clear - or even known - concept. In this section I hope to have provided a solid enough understanding of procedurality, in which resides digital media's unique potential for representation, communication and expression.

The focus of the case study analysis presented here is in how this particular approach for making meaning is realized in the context of game creation. The following section presents the theoretical framework and methodology through which this is done.

3 Framework and Methodology

The case study analyses of the MissionMaker games, which will be presented in the next section, are based on two approaches: the concepts and theories from the field of Multimodal Analysis, and a methodology of my own, developed in a previous research. Each allow approaching the same subject - the expressive and meaning potential of code - from a different perspective.

The Multimodal approach provides the framework to analyze procedurality as a language (or MODE, as we will see), side by side to other languages and artforms, which means dealing with "higher" level functions, such as those present in the software layer, and to the more specific implementations and uses. My own methodology, on the other hand, developed in the context of my previous research, allows analyzing digital media via the "lower" level functions of computational logic and hardware, in a more fundamental sense (FERREIRA, 2011).[15]

In the following pages I describe these two approaches further, as well as their role in the case study analyses.

3.1 **Multimodal Analysis:** procedurality as MODE

One of the main challenges I face on my research is in defining the exact nature of its object of analysis, **procedurality**. It is possible to describe many aspects and identify uses, applications and properties of procedurality itself (as presented in the previous section). However, there are also multiple perspectives through which it may be approached in terms of its role in the creative process.

During the development of this study, I came in contact with the Multimodal Analysis approach (JEWITT, 2014). This is a relatively novel research methodology, which provided an interesting and useful perspective on procedurality. It focuses on **meaning**, and on the different ways it can manifest itself across and between different communicative MODES, or languages. This flexibility allows a broader and more complete framework for analyzing acts of communication, representation and expression, based on what they actually mean or realize, as opposed to being limited to the preconceived models and concepts for each language. Although these preexisting models are relevant, multimodal analysis allows different perspectives which are also important and useful.[16]

Multimodal analysis is particularly useful in the case of digital media, a medium that is still in its early stages of development, if compared to the more mature and consolidated traditional medium and languages. In this study, at first I apply this approach - or rather a simplified subset of its framework - to analyze specific uses of digital media, as a means to access and investigate the language itself, its grammar, structure and potential for making meaning.

One of the main concepts used in this study is that of **semiotic resources**, which comes from the field of social semiotics. It refers to any element (physiological or technological) which may be used to communicate, represent or express an idea, message or meaning potential (LEEUWEN, 2005).

[15]The concept of levels of abstraction is elaborated further in the next section (p. 23). The procedural semiotic resources, which will also be presented in the next section, can be seen as occupying an intermediary position in this continuum - “higher" than the lower level functions of machine language (or hardware, for that matter), but "lower" than specific uses, such as in certain categories or types of creations, or even particular productions.

[16]The term "MODE" is in uppercase to indicate that it refers to the concept in Multimodal Theory.

In the particular context of this study, “semiotic resource" is analogous to the concept of a word in verbal language, considered as a unit inside a sentence - or, in Semiotic terms, a unit inside a syntagm.

The notion of treating the **choice of MODE** as a significant operation in itself is another important idea from the Multimodal approach. This is particularly relevant when discussing how a MODE may relate to other MODES, identifying the resources it shares with other languages, as well as the expressive strategies that are unique to them. This concept also relates to the analysis of the intentions behind particular creative acts, which in turns influences their expressive potential (this is discussed further, in the presentation of my own methodology of analysis).

Procedurality as MODE

In this study, the analysis of procedural authorship and meaning making in the context of the Multimodal framework is done by considering procedurality as a MODE, or language, in itself. This allows the identification of the semiotic resources which are specific to procedurality, as well as the ones shared with other traditional languages and forms.

An alternative way of relating procedurality to the concept of MODE, which is not the one used in this study, is to consider it as a **combination of MODES**. This can also be described as an "orchestrating MODE" (BURN, 2013), which basically operates by organizing and presenting content from various MODES in the same context - spatially, temporally and so on. In that case, procedurality would involve many MODES coexisting and operating through their own mechanisms and logic (resulting in a situation more accurately described as "multimedia"). However, the focus of this study is not in the use of computers as a metamedium (p. 9), but on the meaning potential of the actual procedural expressive strategies available in this medium.

According to Janet Murray, the expressive use of a medium is achieved by exploring its unique and fundamental properties and affordances. The expressive form is opposed to the additive form, which refers to the use of a medium via expressive strategies from previous mediums (MURRAY, 1997). Murray illustrates this with film, which originally was an additive form, based mainly in the use of expressive strategies from photography and theater (such as the structure of the sets and the positioning of actors in the scene, as well as the concepts of composition and framing). After a period of formal exploration, the language of film eventually developed its own expressive identity.

Similarly, the expressive potential of digital media would not come from directly combining elements from previous traditional mediums, but by exploring the properties that are unique to computers. Therefore, in this study, procedurality is not considered to be a combination of MODES, but as a MODE in itself.[17]

[17] There is nothing wrong in using a particular medium as an additive form, or as a metamedium. For example, nowadays most writers use computers for their work - they are not exploring the full expressive potential of digital media, but this is irrelevant in this case, since their form of expression is literature.

Empirical Research

My research has always involved practical analysis of productions, tools, software and creative process of artists and creators, as well as conversations with artists and researchers in the field. However, this has also always been secondary to the theoretical elaborations.

The study presented in this text is heavily based on the empirical analysis of specific case studies from the game design workshop. As mentioned before, this involved the observation of the creative process, analysis of the finalized games (and project files) and conversations with their creators (p. 21). However, following my academic background and research experience, this is still mostly a theoretical and reflective study.

Therefore, Multimodal Analysis is used here in a more general and considerably less detailed manner, if compared to other studies using the same framework. The interviews and analysis presented here should be considered through that perspective. The intention is to identify and describe the usage of procedural semiotic resources in MissionMaker, as a means to investigate and discuss procedurality in a more general sense.

3.2 Methodology of Analysis for Procedural Authorship

In my masters dissertation, which was the result of my previous research, I did an initial investigation on procedural authorship (or "procedural poetics"), defined as artistic manifestations based fundamentally on the programmability and autonomy of digital media.

I also examined how these specific creative practices, due to their aspect of self-reflection, can serve as a glimpse into the artistic thinking, as well as the fundamental role of these procedural strategies of creation in the consolidation of the digital medium as an independent expressive form (FERREIRA, 2011).

Part of that research consisted in defining a methodology of analysis, as a means to identify and examine the manifestations of procedurality in a given production or object (and its different elements and aspects) and in the creative process (and its different stages). This was applied to creative and artistic productions in a wide range of styles and categories, from installation and web art to performance and computer games.

The study presented here makes use of some of the concepts and theoretical tools developed in that previous research.

The methodology of analysis involves identifying the different **influences and contributions** present in a given art production. The framework includes an initial proposal for a classification of the different layers involved in the creative process in digital media. These go from the most fundamental elements and properties of the physical world (including the body and the mind of the artist), to the social and institutional aspects, including science and formal knowledge, the simple mechanical and electronic tools and machines, all the way up to digital media.

This structure, although not meant to be a definitive or comprehensive description, provides a starting point to identify and analyze the different influences and contributions to each stage and element involved in a creative production (both external and internal to the creator or artist). Due to its scope, this classification also allows relating the procedural expressive strategies to other creative and artistic manifestations, and even to other human activities.[18]

The methodology includes a series of **perspectives of analysis** through which the procedural creative and expressive strategies can be investigated, to be applied in parallel to the aforementioned classification of layers of influences and contributions. This allows an objective description of the elements involved in each one of the different stages of the creative process.

Among these different possible perspectives is the consideration of the different individual elements and properties of a certain production (or in a certain creative process), as well as the influences and relationships between them, and their role in the artist's creative or poetic intention. For example, the expressive qualities of a work of literature are not influenced by its physical format of presentation, at least not as much - or in the same way - as in sculpture. In the context of digital media, and particularly in computer games, this approach allows identifying how the procedural elements of the system contribute (or not) to the expressive qualities - or meaning potential - of a particular production.[19]

Another perspective of analysis is to consider the different stages of the creative process, from the initial insight or idea in the creator's mind, going through the conceptual and practical tools used and finally the end product in itself (be it an object, a software, a performance and so on).

In the case of procedural authorship, it can be found (or applied) in three different moments in the creative process: (1) it can be an intermediary step in the creation of the final product (such as in most types of generative art - for example, an algorithm that generates a static image); (2) procedural strategies may be used as an auxiliary tool in the creative process, but not directly in the creation of the final output (for example, a dance performance based on a choreography composed via procedural tools); or (3) the output itself may be procedural, which is the case of any software-based production, such as digital games.[20]

This methodology may also allow mapping the creative or poetic intentions behind a particular production. In the case of digital media, this is a specially complex process, since there are many different elements, aspects and stages, as well as contributions, involved - this topic is elaborated further in the next section, when discussing meaning in digital media (p. 26).

[18]As mentioned before, despite the potential complexity of such an endeavor, the intention of this classification is not to provide a definitive or complete description, but to allow a framework on top of which further analysis and reflections can be developed.

[19]Of course, creators and artists are free to define their own approach to a particular language, which may lead to interdisciplinary (or multimodal) endeavors, or even to novel forms and languages. The proposed analysis still stands, and allows identifying and understanding these particular manifestations.

[20]The separation of the creative process in these three stages is not rigid or absolute. For example, the third situation described in this paragraph could be considered as a particular instance of the first one, considering a distinction between the execution of the software and the algorithms themselves that compose the software (textual, and therefore essentially static). This topic is also discussed previously in this text, when presenting the common misconceptions related to procedurality (p. 12).

The concept of Process Intensity (or Procedural Intensity) proposed by game designer and theorist Chris Crawford, mentioned previously in this text, is also useful in this methodology, since it allows referencing the proportion of procedurality in a given digital content, in relation to static data (CRAWFORD, 1987).

We have seen that the difference between procedural content and static data is essentially arbitrary, and that procedural content will always include static elements in their fundamental composition or structure. Therefore, the measure of Process Intensity must also be considered when analyzing the meaning potential of procedural elements (or semiotic resources) in a digital production. This allows a clearer understanding of the languages (or MODES) involved, as well as the different types of expressive strategies available (from the perspective of the creator or designer) or the repertoire required for its fruition (in the case of the audience or user).

This section presented the main concepts and theoretical tools which were used in the case study analysis, either indirectly, as a guide to identify relevant aspects and elements to be considered, or in a more direct and practical manner. The analyses themselves are presented in the following section.

4 Case Study Analyses

The object of the case study analyses presented in this section are the computer games created in the context of the Playing Beowulf pilot workshop, as well as the creative process itself, including the MissionMaker software. The objective is to identify and analyze the use of **procedural semiotic resources** by the creators of these games, as well as the availability of such resources in this particular software, as a means to discuss and investigate the **meaning potential** of procedurality in a more general sense. The framework used, as described in the previous section, is the Multimodal Analysis approach, as well as a methodology of my own.

In the Playing Beowulf pilot workshop, student teachers from the English and Drama programmes were encouraged to explore the poem Beowulf via a series of creative exercises in a variety of different languages and mediums, including drama, improvisational theater, creative writing, drawing, animation, film and, finally, digital games, using the software MissionMaker (Fig. 9, below).

Creative Writing
Drawing
Animation
Film
Drama
Improv. Theater
Digital Game
(MissionMaker)

Figure 9: Workshop: languages and mediums explored

Each participant (or group) in the workshop was instructed to use MissionMaker to create a game based on a particular section or situation from the poem. The objective was to explore game mechanics as a representational medium in itself, through its own unique expressive strategies.

The version of MissionMaker used in this study was developed before the Playing Beowulf project, and therefore the software features a more general set of tools and assets to the users. At the time of this writing, there is a new version currently being developed, which is customized to this particular project, including characters, objects and settings, as well as behaviours related to the Beowulf story and poem. For example, there is a specific character creator system being developed for Grendel and his mother, as well as a sword fighting mechanic. A different interface is also being implemented, to fit with the particular workflow expected for the software's usage in schools, museums and exhibition settings.

The Playing Beowulf pilot workshop involved two days of activities. The first day included all of the aforementioned traditional creative exercises, while the second day was dedicated exclusively to the game creation activity.

According to several of the participants, dealing with the more traditional (and familiar) languages and mediums first allowed them to get familiar with Beowulf's story world, narrative and characters, as well as to build their own image and perspectives about the original text. That way, on the second day they could focus more attention on learning the MissionMaker tool.

Of particular interest to this study are the intersections between the traditional activities and the game design exercise. As we will see further on in this section, by considering the shared semiotic resources between the creative process in using MissionMaker and in the other exercises in the workshop, it is possible to better understand the particularities of procedural authorship in relation to other traditional languages and forms.

In this section I first present the **Procedural Semiotic Resources** identified in the MissionMaker software, then I discuss the process of **Composition**, through which the resources are combined to form the finished game. Finally, I develop further how the process of **Comparing MODES** may help to further investigate the particularities of this process, by identifying and comparing shared semiotic resources between game design process in MissionMaker and the other creative activities from the Playing Beowulf pilot workshop.

The analysis of the games involved the following steps: observing the creative process during the workshop, examining the finalized projects (as well as their files in the editor) and interviewing the creators. The games were selected for the analysis according to how well they reflect their designer's understanding of the tools and functions provided by MissionMaker, as well as their level of completion, regarding structure, functionality and amount of detail.

After the workshop, I played the games, and analyzed them, as well as the project files in the editor. Individual meetings with the creators of the games were then held, in which we played the games and discussed their experience in the workshop. This process was documented in two articles (FERREIRA, 2014a and 2014b).[21]

[21]It should be noted that all of the participants featured in this study are female, except for one. Although in the workshop there were some male participants, they were very few.

4.1 Procedural Semiotic Resources

The concept of **semiotic resources** is associated with the communication, representation or expression of **meaning potential** (p. 16). Since this study is interested on the meaning that is expressed through **procedural** means, the focus is on the "**procedural** semiotic resources".[22]

As explained previously, procedurality can be defined in relation to **static** data (p. 6). While static data represents information directly, via words, numbers, tables, images, sound and so on, procedurality represents rules or behaviours. While traditional creators use static elements in their work, artists exploring the expressive potential of digital media use procedural strategies - in a recent interview, artist Casey Reas, co-creator of the programming language Processing, declared: "I work with statements, variables, loops, conditionals, functions, objects, and arrays." (REAS, 2015)

A straightforward example of a procedural element in MissionMaker is a rule that says that a door must open if the player touches it - this is a dynamic behaviour. The actual door object, on the other hand, is an example of an element that is not procedural - it is static. However, as mentioned earlier, this distinction is not always so rigid, specially from the perspective of semiotic resources, as we will see later in this section.

In the following pages I introduce some concepts of computational logic, as well as some other relevant initial elaborations. I then present the procedural semiotic resources identified in MissionMaker, which can be used by the designer to exercise procedural authorship. Finally, I also present the particular set of resources that are exclusive to the software's engine.

Initial Elaborations

Levels of Abstraction Digital games are a particular type of computer program. As such, they are compiled from algorithms, which consists of a series of instructions that represent a particular **abstraction** of a process, much like a map is an abstraction of a place. A same place can have many maps, each one providing a certain view or perspective. One may emphasize the different types of terrain, another may present borders and some other may even show an artistic subjective interpretation of that same place.

Algorithms are the same, in the sense that a single subject may be represented in a variety of different ways, according to the particular perspectives and purposes of the designer (or programmer). However, unlike a map, a computer program allows the representation of systems, rules and behaviours, in a procedural manner.

There can be many **levels of abstraction**, counting up from the original object or concept that is being represented. So, in the example of the map, considering the actual location as the original object, a low-level abstraction could be a realistic 3D representation of the region, a medium-level abstraction could be a typical 2D map (with roads, borders and so on) and a possible higher-level abstraction could be a pie-chart representing the proportions between different kinds of terrains - note how, as the "map" became more abstract, direct elements and aspects from the original object were

[22]Following the analogy with verbal language, the building of the sentence itself (which happens within a particular Grammar - in this case, specific to MissionMaker) would be the act of **composition**, which will be presented further on in this section (p. 42).

removed or diminished, while certain more specific elements or aspects were emphasized (the last "map", for example, has no information of the actual shape of the terrain, but it contains potentially useful information that is not present in the previous maps).

In digital media, the lowest level of abstraction is usually considered to be machine language, which gives access to the hardware logic. Going up, there are the higher level programming languages, which provide additional abstractions for dealing with machine language, and eventually the application software, which provide even higher levels of abstraction, usually through advanced visual interfaces. On one hand, each of these new levels of abstraction provide a particular type of view and access to the computer's internal mechanisms. On the other hand, they also limit access to the levels that are underneath. Levels of abstraction are not a rigid definition, and there are no clearly defined boundaries between what is considered to be a low or a high level of abstraction (or any intermediary levels, for that matter).

The concept of levels of abstraction will be useful throughout this section, for describing and analyzing the different kinds of procedural semiotic resources in MissionMaker.

Executing Meaning Procedural semiotic resources are always represented as code (or algorithm) in the memory of the computer. However, the actual resources are not the pieces of code themselves, but their effect in the execution of the program, or system. In other words, the meaning potential in a particular procedural semiotic resource is not its specific implementation in a certain program, line of code or function, but in its design, or logic. This is apparent by considering that a same procedural semiotic resource (such as a rule or behaviour) may be implemented in multiple ways, using a variety of different programming languages, approaches and techniques.[23]

As mentioned before, a certain procedural semiotic resource can only produce meaning in the context of the execution of its code. This can be directly, through the materialization of some kind of output that can be perceived by the user or audience (as shown further on in this section), or indirectly, by influencing other functions or procedural semiotic resources contained in the system or program.[24]

This reflects a particular property that differentiates the procedural semiotic resources from the static ones. Both share the social aspect, in the sense that their meaning potential can only be realized in relation to a given receptor, in a particular context and so on. But the **procedural** semiotic resource may depend on an additional aspect, of technological nature, which is its integration within the system's logic. In other words, a certain semiotic resource can only be procedural if either it is itself procedural, or if it affects the logic of the system in some degree.

For example, in MissionMaker, any audio or textual content added by the designer to a game is a **static** semiotic resource. It is static because their actual content has no influence over the computational logic of the system. That is, any particular audio or text used in the games could be swapped with a different content and the execution of the program would be identical. They may

[23]In practice, the very choice of what approach to take is usually meaningful in itself.
[24]Assuming the program produces an output or effect that can be perceived externally, in real-time or not.

still carry meaning potential - for example, in most of the games that were analyzed the characters deliver important narrative elements via audio or text content. But these are not **procedural** semiotic resources, since they don't participate in the logic of the system, and therefore, in this case, they can't provide meaning potential of procedural nature.

However, consider if MissionMaker provided a function for analyzing audio, even a simple one, able to detect relative volume. Suppose this could allow the game designer to create a rule that makes certain characters attack whenever they hear a loud sound. With that, audio files could be considered a procedural semiotic resource, since they would be participating in the logic of the system. The resource itself (the audio) never changed, but the context (system) changed, via the addition of the function able to analyze audio. Also, in this case, although the audio itself is now contributing to the meaning potential in a procedural sense, it is still in a lesser degree, compared to a rule or behaviour, for example (such as the function able to analyze it).[25]

Another point which is relevant to the analysis presented here is the fact that, while procedural meaning can only be realized through the execution of the code in which it is represented, this doesn't always require user interaction to happen. In the particular case of computer games, interaction is required in order to realize the meanings represented specifically in **gameplay** elements. For example, in the classic computer game The Oregon Trail (MECC, 1971), different aspects of the life of 19th century American settlers are represented via systems, which need to be interacted with in order to be realized.

On the other hand, many games also present meaning via **"non-gameplay"** elements, which are the ones not necessarily dependent or associated with gameplay in order to be realized. For example, most real-time simulations reveal many aspects of their systems even without being interacted with. By simply observing the characters and elements following their behaviours and rules, it is possible to grasp the meanings represented in the system, like the audience would in an improvisational theater performance, for example.[26]

As mentioned before, in the case of computer games, interactivity is required in at least in some level. However, regardless of the necessity of interactivity for the realization of a particular procedural meaning, in this study the focus is not on the interactions themselves, but on the procedural systems (or algorithms) through which they are made possible.[27]

In order to further contextualize the procedural semiotic resources that will be presented in this section, it is also useful to understand in practice how they relate to the actual execution of the code. A computer program is usually organized in two main parts (or sections): the "setup" and the

[25]The idea of measuring levels of procedurality is related to the concept of Process Intensity, mentioned previously in this text (p. 19).

[26]Additionally, the final product may not even be an executable program, but the result of its execution, which is the case in many generative artworks, such as the images created by the aforementioned "digital painter" Aaron (p. 6). The execution of the code is still indispensable for the materialization of the artwork, although in this case the realization of the actual expressivity (or meaning) in the artwork only happens when the images are viewed by an audience.

[27]The topic of interactivity is also discussed previously in this text (p. 10).

"loop" - they are essentially the same in the sense that both may contain code, but they differ in their purpose in the context of the execution of the program. The "setup" is executed only once, before the "loop", and it usually contains code related to preparing the environment for the execution of the rest of the program - this may include the definition of parameters for the outputs (visual, audio and so on) and inputs, and global variables and functions. The code inside the "loop", on the other hand, is repeated indefinitely.[28]

What this means in a practical sense is that, in principle, all of the functions, attributions and processes contained in the “loop" section are "alive" (always updating). From the perspective of the procedural semiotic resources, presented later in this section, this means that any function or operation may be used in a program in a way such that it is repeated (indefinitely or not), even those that don't originally involve loops (such as a simple variable attribution, for example).

Meaning in Digital Media: Multiple Contributions When discussing expression and representation of meaning in the context of procedural authorship, it is particularly relevant to consider the origin and intention behind such acts. This is because, in digital media, there is a significant number of layers and contributions involved in every process, as seen previously (p. 18). For each expressive or significant act or production, there may be a wide range of contributions - from the actual creator, to the programmer of the software and tools, as well as the designers of the hardware itself.

Traditional tools and processes may also involve a variety of layers and contributions. The brush used by a painter has a particular design which influences the artwork, for example. However, in digital media the systems are usually much more complex. For example, in a digital painting, although its expressive qualities are usually the result of the work of the artist, the designers (or programmers) that created the brushes and effects that were used may also have a significant influence in the final product.

In MissionMaker, although the software offers a relatively high degree of customization to the user, this is mostly limited to static elements and properties. As we will see further in this section, procedural authorship in MissionMaker is restricted to mainly three aspects: the attribution of variables, the creation of rules - or conditional statements - and the choice of composition. All the other procedural semiotic resources are locked into the game engine.

This means that most of the **procedural** semiotic resources involved in a typical MissionMaker game are authored by the creators of the software, not the designers of the games themselves. However, this doesn't always necessarily translate to a diminished creative control of the game designers over their creations. Specially considering the subjective nature of expression, representation and meaning, both from the perspective of the creator and the audience - for example, a particular semiotic resource may be used by one creator to express an idea in a meaningful way, while a different creator may be unable to represent any meaning whatsoever with the same resource.

[28]This is a simplified generalization, used in order to explain the basics of the execution of a computer program. These two sections mentioned are not necessarily always present, and different programming languages and approaches may have different nomenclatures, or even additional categories.

In my analyses of the MissionMaker projects, I found that most of the authorship exercised by the game designers happened via static elements, such as text, images, sound files and music. There were still, however, several instances of procedural authorship, which will be presented further on.

Choice of Perspective of Analysis The procedural semiotic resources identified in this study were selected as being the most relevant to the game designer using MissionMaker. Therefore, this is not meant to represent a comprehensive listing, neither in regards to digital media, nor to games in particular. A different object of analysis, such as a different software, or a low-level programming language, could lead to the identification of a different set of procedural semiotic resources, organized in a particular structure of its own. Additionally, the categorization presented here is not intended to be rigid or definitive. It represents initial considerations, which are flexible and open to future revisions.

There are many semiotic resources in MissionMaker that are not accessible to the user, which are part of the software's engine. In the end of this section I briefly present some of them, explaining how they can affect and contribute to the execution and displaying of the games developed in MissionMaker, both in a technical sense and in terms of their meaning potential.

The main procedural semiotic resources in MissionMaker which are available to the user are the **variables** and the **rules** (or conditional statements). As we will see, variables, besides being a semiotic resource in itself, can also be part of conditional statements. This kind of nested hierarchy can be illustrated via an example presented by Theo Van Leeuwen in his introductory book on social semiotics, in which the act of walking is described as a semiotic resource (LEEUWEN, 2005).

Leeuwen explains that, since a person can walk in a variety of different ways, there is a potential for meaning (or expression) in this action. A slow, methodical walk style may express fear, while walking with a swing may express joy. However, one could also identify a series of other semiotic resource subsets, that are contained in the greater semiotic resource of "walking", such as "leg movement", for example. A person can be sitting down and, simply by moving or positioning their legs, express a variety of ideas and meanings (crossing the legs to express formality, opening them to express intimacy, shaking them to reveal nervousness and so on).

Some of the meaning potential available in the subset of "leg movement" may intersect with the greater semiotic resource of "walking", while some may apply only to this particular subset. The point is that each one of these choices of perspective allow a different way of approaching the same subject, through different scopes and levels of detail, and for particular purposes.

In a similar manner, the procedural semiotic resources that will be presented in this section also come from different levels of abstraction. As mentioned before, these go from the lower level, featuring elements such as variables and data structures, up to the level of the conditional statements and functions, and finally to high-level abstractions, such as the conceptual models. Since the focus of this text is on the semiotic resources available in MissionMaker, not all of these abstraction levels are presented here (the lowest levels would be machine language, or perhaps the hardware logic), and not all of the possible procedural semiotic resources for each one of the levels is present.

Also, in MissionMaker, procedural authorship may happen in certain elements or levels of abstraction, but not in others. For example, the software does not provide the user with tools for modifying the movements allowed for the characters, such as how they turn their heads to follow another character. This means that this is not a semiotic resource available to the designer, since there is no creative control over it. On the other hand, the designer does have access to the rules that define in which circumstances that particular behaviour will be triggered. That is, in this case, the designer doesn't have creative freedom in a lower level (that particular behaviour), but there is some control in a higher level (the triggering of such behaviour).

4.2 Procedural Semiotic Resources in MissionMaker

The previous pages presented the concept of procedural semiotic resources, as well as some initial considerations regarding them in the context of digital media. The following pages consists of the actual analysis of the MissionMaker software, considering the perspective of the designer using its tools and functions in order to create a game.[29]

First, I will briefly present the user interface (UI) of the MissionMaker editor (Fig. 10, below). However, rather than describing the complete structure of the software itself, including all of the available functions and features, I will present it from a more practical perspective, considering the workflow of a game designer approaching MissionMaker with a particular idea or creative intent.[30]

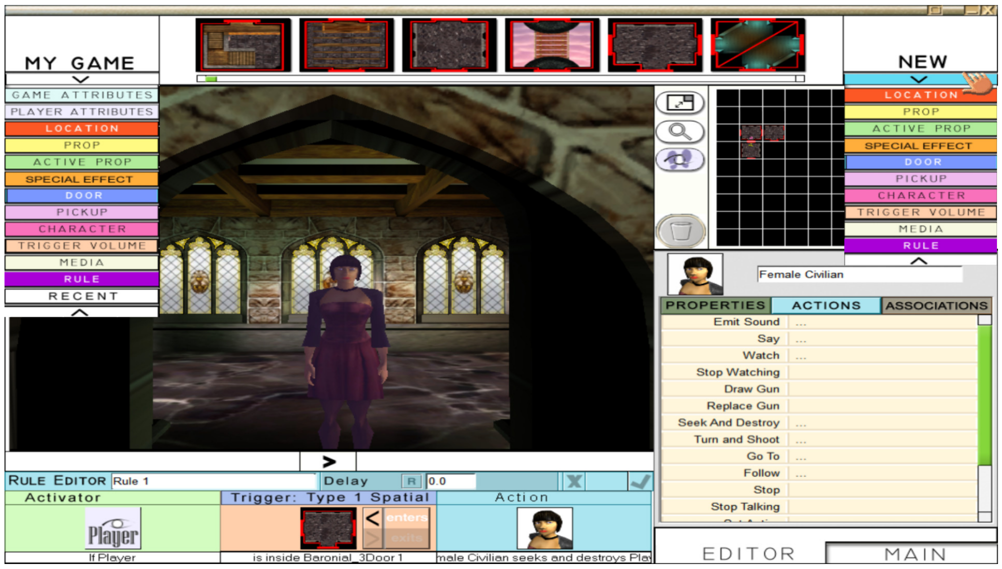

Figure 10: MissionMaker: user interface

After creating a new project and entering the editor, most users begin creating their game by adding **locations** to the map in order to compose the layout of the level. This is done by dragging the icons representing the desired locations from the visual menu on the top, down to the map view.[31]

[29]This is based on the observation of the participants working on their games during the workshop, as well as my own experience with the software.

[30]The participants of the workshop received a few instructions about how to use the software before the game creation exercise. They also had help from the tutors, including myself, throughout the activity. For the analysis presented here, the focus is on the actual steps taken by the designers to create their games, regardless of the learning process for using the tool. Therefore, topics related to that aspect of the workshop, such as the usability of the tool itself, for example, are not the focus of this analysis. For a more complete documentation about the software's UI and functionalities, refer to the MissionMaker Teacher Support Packs, in the Immersive Education website - URL: `www.immersiveeducation.eu`

[31]No other physical object, such as Props and characters, may be added until there is at least one room for them to be placed on (this reflects MissionMaker's emphasis on the physical aspects of the simulation, which is discussed further on in this text). The map view can be presented either in the large main viewport, in the center of the screen, or in the smaller viewport to the right, which is the case in the screenshot presented here (the locations can be seen in the grid). Both viewports may contain either the map or a first-person perspective of the environment, which roughly simulates the player's experience during gameplay. The image in this page is a composite of multiple states of MissionMaker's UI (version 2.0 Build 5), representing most of the elements mentioned in the text.

Alternatively, instead of beginning by creating rooms, the designer can also start by creating or editing some other aspects and elements of the game. For example, it is possible to customize the Game and Player Attributes, to edit the existing default Rules and Media, that are present in every new project, or to create new Media (I will elaborate more on these items further). However, in the MissionMaker workflow, these elements are usually dealt with further in the game creation process, and after some rooms and objects have already been created.[32]

After that, most users start including **objects** in the environment (they can be accessed via the NEW menu, on the top right). This usually involves adding Props, Active Props, Doors, Pickup items and Characters. Additional Locations and Special Effects may be included as well. For each type of object selected, a secondary visual menu appears across the top of the screen, showing thumbnail visual representations of the available objects. The process of including the objects in a particular position in the game word is similar to the one described earlier for adding locations. This is done by dragging objects down to the viewport which is currently showing the first-person perspective mode. The object can then be adjusted and modified spatially, directly on that viewport, and the individual Properties and Actions for each object may be accessed via the panels occupying the lower half of the screen, on the right (Properties and Actions are presented as tabs in the same panel).[33]

There are different properties available for each type of object - for example, characters have properties for appearance, strength and walking style, and energy items has the Health level that it will provide if used. Also, each type of object allows for different authoring affordances to the designer, such as in freedom of movement, scale and rotation. Considered as semiotic resources, each object may allow for different uses according to the context - for example, Props are objects with mainly decorative purposes, that may be used as a static semiotic resource, however, as we will see further, they may also be used as procedural semiotic resources depending on their role in the system.

At this point of the creative process, users usually also create some **Media** items, which can be pop-ups (screens displaying text, image or both), audio, video or speech. Media items are a type of content that, differently than the objects mentioned before, don't have a direct physical representation in the game world. As such, they depend on other elements, such as objects or rules, in order to be included in the game system, and to be displayed or presented during gameplay. For example, a Pop-Up can be set to be shown if the player picks up a certain object, or a certain audio file can be played when a character is shot - this kind of behaviour is possible in MissionMaker via rules, which are introduced next.[34]

[32]Game Attributes include variables related to the game's overall system (such as interpretation of user input), and Player Attributes includes variables related to the player's character (such as Health and parameters related to user input). Both items are accessible in the MY GAME menu, on the top left.

[33]To access the Properties and Actions panels for an object, it must be selected (this is done either through the viewport, or via the MY GAME menu). An additional, more advanced, object type available is the Trigger Volume, to be used in association with rules (this will be mentioned further).

[34]The creation of Media items, similarly to the objects, is also done via the NEW menu. Although Media items don't have a direct physical representation in the 3D environment, they may be associated indirectly to particular locations or objects via the game's logic (this allows, for example, to have a certain sound effect play whenever the player goes through a particular place).

Up until this point in the game creation process, the designer only created and modified **static** elements. The following steps describe the process of creating **procedural** elements, in order to describe behaviours or rules.

A rule is defined in a somewhat "inverse" logic, by first choosing the desired resulting Action, and then the condition that will Trigger it. For example, the designer can create a rule which states that a certain character will say something (the Action) as soon as the player enters a certain location (the Trigger). Or a door can be set to open (Action) whenever a certain amount of time has elapsed (Trigger).[35]

Interestingly, however, although there is an item for adding Rules in the NEW menu (in the top right of the screen), it doesn't actually have a function when clicked. This is because, as mentioned before, the rule creation process is "inverted" - the designer needs to first select the action that should result from the rule. For example, in the case of the door opening, illustrated in the previous paragraph, the designer would have to select a door, access its Actions panel and click on the Open action in order to create a Rule that will result in this.[36]

The creative process from that point on differs more greatly from user to user. However, it usually involves going back and forth between the functions mentioned before, in order to build the rest of the game world and the rules. This generally also includes an introduction and an ending to the game, which is usually presented via pop-ups, and sometimes through audio or video content.

As mentioned previously, this is not meant to be a complete description of the functionalities in MissionMaker, but an overview of the creative process and the tools that are available to the users. It shows, for example, that in this software there is a great emphasis in **static** elements. The main **procedural** semiotic resource available to the designer is the Rule, although variables can also be explored in order to contribute with procedural meaning potential, as shown in the following pages.

[35]MissionMaker has several types of Triggers available (Spatial, Global, Speech and so on). A Rule may also have a parameter called an "Activator" - for instance, in the first example presented in this paragraph, the Activator is the player, while in the second example the activator is the passing of time (however, often the Activator is integrated within the Trigger parameter).

[36]After that, the designer still needs to define the remaining parameters of the Rule, via the Rule Editor panel, at the bottom of the screen.

Variables

Variables are one of the most fundamental elements in most computer programming languages, and also an important resource in the context of MissionMaker. A variable is an abstraction that may contain, be associated with or point to different types of data and content, as well as other variables and expressions (which may be very simple or extremely complex). As such, a variable may contain static or procedural content - and, as we will see, in both cases it has the potential to be a procedural semiotic resource.

A variable is usually named in an intuitive manner by the programmer, making a direct reference to what it represents - for example, a variable containing the size of an object may be called "objectSize". In practice, however, this normally doesn't affect the actual logic of the program. Therefore, the naming of the variable doesn't contribute to the system in terms of meaning potential, at least in principle. The actual meaning potential in a variable resides on the designer's choice to include it in the system, how it affects the system and what is being represented in that context.

As mentioned previously, variables may be considered a procedural semiotic resource even if they contain or point to **static data** or content (such as strings of text, numbers, images and so on), providing they affect the system's logic (p. 25). For example, in MissionMaker there is an object representing a wine bottle, which can be picked up and used by the player character. This object has a variable associated with it called "Health", which indicates how much the player's health should increase when it is used (that is, when the player drinks the wine). Although this is a static information (a numeric value), it affects the behaviour of the simulation - for example, it changes how much damage the player character can take in combat. A more straightforward example of this would be the variables associated to the player character itself, mentioned previously.

In the context of **MissionMaker**, variable attribution is a procedural semiotic resource with a limited meaning potential. There are two reasons for this. First, because it is not possible to directly create new variables. It is possible to create new entities, which in turn contain variables associated with them, but it is not possible to add or remove variables from these entities. Second, because only a certain set of the variables used in the simulation may have their values modified, or used as parameters for the definition of rules and behaviours.

These restrictions also have an important implication regarding authorship, since these predefined procedural semiotic resources represent meaning potentials that cannot be changed or modified by the user. For example, most of the variables available emphasize the physical aspects of the simulation. The variable "resistance" defines how much damage a character can take, "scale" defines its size in the game world and so on. The game designer may choose different values for these variables, and thus exercise a certain degree of authorship, but it is still very limited. Although new variables and economies could be implemented through some clever manipulation of the entities and variables available in the software, it still holds true that there are several meanings and models which are predetermined - or hardcoded - in the tool.[37]

[37]This limitation is even more significant in the case of game variables that are not modifiable by the user, or those that can't be referenced in the rules and behaviours (this is the case of the resources in the software's engine,

Most of the constraints in MissionMaker exist by design, as a way to make the game creation process more accessible and straightforward to the user. The simplified options and modular approach allow a much more intuitive and straightforward workflow, making it easier to quickly build fully functional games (specially to young children, and inexperienced computer users).

Having said that, limitations in the tool are not necessarily an obstacle, specially considering that often creativity thrives under constraints. For example, in one of the games analyzed in this study, designed by Andrew Smith, he included the wine bottle, mentioned previously, but the "Health" variable was modified to a high **negative** value. This resulted in damage when the player consumed the item. By doing so, Smith was adding a mechanic that punished the character Beowulf (controlled by the player), if he got distracted and decided to drink wine instead of following with his mission (in this particular case, the procedural meaning in this mechanic was also reinforced by a static semiotic resource, in the form of a pop-up text message reprimanding the character).[38]

presented further - p. 38). Even the default values for these variables, although modifiable, they reflect the particular perspective that the designers of the software had in relation to these resources.

[38]The focus of this study is in the analysis of the games and their internal logic, not of the MissionMaker software itself, its editor or game engine (including the internal variables and all the other procedural semiotic resources that are not directly related to the logic of the games themselves, such as the parts of the program related to the interface, data handling and so on).

Rules: Conditional Statements

Conditional statements are functions that, combined with variables and expressions, control the flow of execution of a program according to certain conditions. This is one of the main elements that allow procedurality to happen in digital media, since the conditional is an inherently procedural semiotic resource. In MissionMaker, conditional statements are available in the form of "Rules", which are the main resources for procedural authorship in this software.

One of the most straightforward and practical advantages of using conditional statements is that they allow and facilitate the creation and management of more complex rules and systems (as compared to simple variable attributions, for example). However, perhaps the most important aspect of this resource is that it provides an additional layer of abstraction in the logical flow of the algorithm. This may be explored by a designer, creator or artist to represent or express meaning potential in a more robust and sophisticated manner.[39]

As mentioned before, in **MissionMaker**, among all the procedural semiotic resources available, conditional statements are the ones that allow the designers to represent meaning potential with the highest procedural intensity (p. 19), at least in principle. The designer has a relative high degree of freedom for creating and defining new rules, or behaviours, which can be associated and related not only to characters, but also to most objects and other aspects of the simulation and the game's logic.

However, there are still limitations in how much the designer is able to express through this particular resource in MissionMaker. For example, although there are many options available for Triggers, only one is allowed for every rule. This means that it is not possible to create more elaborated rules, requiring more than one condition to be met. Also, each element, such as an object, a character or a room, has a fixed set of actions that can be triggered by a rule.

As with the variables, the amount of meaning potential that can be represented through this particular semiotic resource is determined by how much control the designer has over its creation and modification. In the particular case of conditional statements, this may also vary according to the level (or layer) and element being considered. On one hand, elements which are hardcoded, such as the predetermined set of triggers mentioned before, don't allow the user the opportunity to represent or to make meaning. On the other hand, aspects that are not fixed (or less limited), such as the possible targets available for the rules, allow the user a greater control.

As mentioned before, this also depends on each particular creator, their intention, the audience and context in which each semiotic resource will be used. For example, one designer may require a certain feature that is not available in a particular semiotic resource - therefore, from the perspective of this user, the software is very limited. A different designer, however, may consider the same feature to be irrelevant.

[39]In theory, in some cases a conditional statement could be implemented using simple variable attributions being executed in the program's loop. In a general sense, often complex higher-level functions can be substituted by alternative solutions based on more fundamental and simpler elements. The concept of levels of abstraction was presented previously (p. 23).

Compared to other game authoring applications with a similar target audience in mind, MissionMaker offers a relatively powerful set of tools for creating rules and behaviours. However, in most of the games analyzed in this study, rules are used in a superficial manner, for simple and straightforward tasks - for example, triggering a text or audio message when the player clicks on a character, or opening a door when the player picks up a certain object. Also, few of the games explored rules and behaviours as procedural semiotic resources, taking full advantage of them in order to create procedural meaning potential.

The following are two examples in which the designers were successful in expressing a particular idea or meaning via rules and behaviours.

The first example is from the aforementioned game by Andrew Smith, a teacher student in the English and Drama programme. In his game the player adopts the role of Beowulf, the title character of the poem. In the beginning of the game the player encounters the king, who requests him to kill Grendel's mother and find his crown. The game ends in victory for the player if the beast is killed (Fig. 11, below).

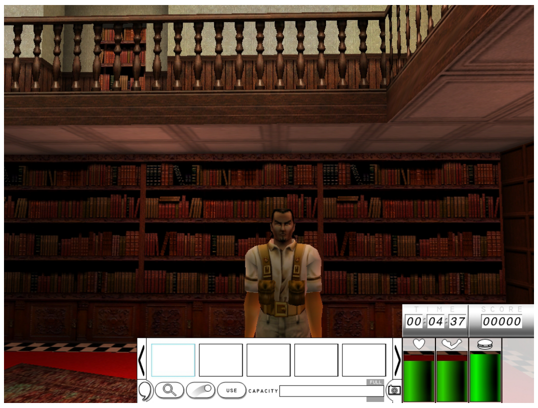

Figure 11: The King (Andrew Smith)

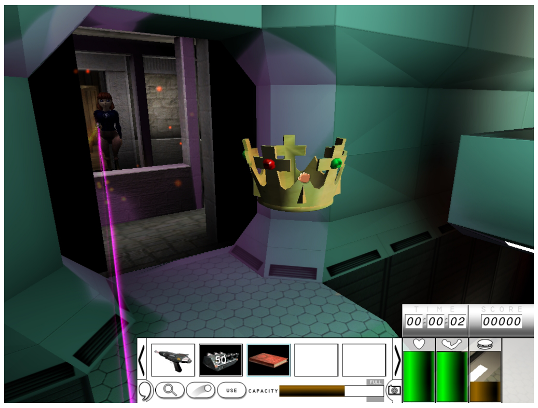

Figure 12: Mother attacks (Andrew Smith)

In this game, Grendel's mother attacks if the player picks up the crown which is located in her lair, or if she is attacked (Fig. 12, above). The rules in MissionMaker are as follows:

```
If Crown becomes Owned by Player
then Grendel's Mother Seeks and Destroys Player.

If Grendel's Mother is shot by player
then Grendel's Mother Seeks and Destroys Player.
```

Smith was representing two different levels of meaning through these rules. First, he provides a straightforward reaction to the theft of the crown, as well as to an eventual attack by the player. However, the behaviour is also meant to fit into the broader perspective of the story told in the original poem.

In the story, it is not necessarily clear who is the "villain" and who is the "hero". From the perspective of Grendel's mother, Beowulf is the "monster" who killed her son. By having a behaviour associated to the character of Grendel's mother, which only makes her attack the player if a certain boundary is crossed, Smith is representing the fact that this character is not inherently "evil", therefore echoing the ambiguity of the original story. In other words, Smith used a Rule, which is a procedural semiotic resource available in MissionMaker, to represent a particular Meaning Potential.

Another example is in the game created by Tracey Matthews (Fig. 13, below). In her game, the player, in the role of Beowulf, can speak to some of the warriors by clicking on them.

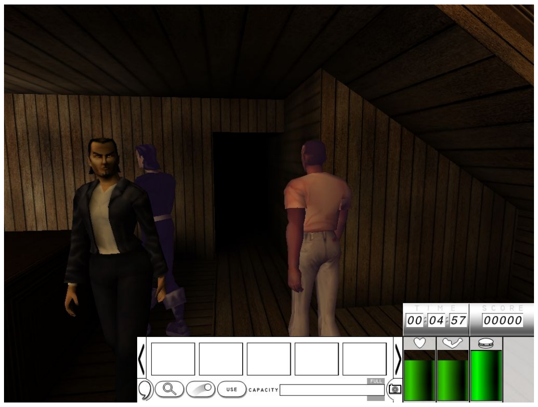

Figure 13: Warriors (Tracey Matthews)

One of them, after spoken to, begins following the player around. The rule is:

```
If Warrior is clicked then Warrior starts following Player.
```

This rule was implemented by Matthews as a way to represent the loyalty of the warrior towards Beowulf, who is considered by them as their leader in this context.

This meaning (the concept of "loyalty") does not result solely from that rule, but also from its presence on this particular system, and its relationship to the other semiotic resources, both procedural - such as the actual following algorithm, for example - and static - such as the popup text message containing the dialogue associated with that situation. The same rule could be used in a totally different game, but being used to represent a different meaning potential (or no meaning at all).

Interestingly, the specific behaviour that allows a character to follow another in MissionMaker is hardcoded in the game engine. Therefore this particular behaviour is not a creation of the game's designer, but of the software's creators. On the other hand, it was Matthews' choice to create the rule that made this behaviour be triggered by the player interaction with that character. By doing that, she leveraged the meaning potential of a preexisting behaviour - the algorithm for "following" -, integrating it into her own rule.

This rule in Matthew's game also illustrates well the idea of breaking from the objective or systematic aesthetics, which is common to procedural expression, as mentioned previously (p. 11). Although the behaviour results from a precise rule, the way it is integrated into gameplay makes the actions of the NPC seem organic, almost life-like. During the workshop, when other participants played Matthew's game, most of them didn't even notice the behaviour at first. After interacting with the game for a while, eventually it would become clear that the warrior had been following Beowulf since they spoke.

This belated effect is only possible because in this case the system doesn't respond directly to the player's input, which is what happens in the rule in Smith's game, mentioned before. As soon as the player picks up the crown, Grendel's mother begins attacking, making it explicit to the player that there was a rule behind this behaviour.

Once again, these behaviours are both also influenced by hardcoded functions from MissionMaker's engine. While the behaviour for following is relatively organic by design, the "seek and destroy" behaviour was implemented in a much more straightforward manner. These predefined functions from MissionMaker's engine will be discussed next.

MissionMaker Engine: Hidden Meanings

The two procedural semiotic resources presented previously - **variables** and **rules** (or conditional statements) - are the main elements identified in this study through which the game designer is able to exercise procedural authorship in the context of MissionMaker. There are also many other procedural semiotic resources that are an integral part of the games created using this tool, but that are not accessible to the user.

This section will present some examples of these resources, from different levels of abstraction, analyzing how they affect and contribute to the execution and presentation of the games, both in a technical sense and in terms of their meaning potential.[40]

As explained previously, levels of abstraction in digital media represent the distance between a particular abstraction and the more fundamental logic of the medium (p. 23). For example, considering the procedural semiotic resources available to the designer in MissionMaker, presented previously, variables have a lower level of abstraction, compared to conditional statements (or Rules).[41]

For the purpose of the following examples, **high-level abstractions** refer to the functions that are specific to MissionMaker's engine, **medium-level** refers to the programing language used to design the software and **low-level** relates to the computational logic of digital media in a more fundamental sense.

The analysis on the higher-level abstractions is more thorough in comparison to the ones for the levels below, since lower level abstractions are usually more generic in terms of function, and therefore less specific to particular software and applications.[42]

High-Level Abstractions

The most evident high-level abstractions in MissionMaker's engine are the Rules and Variables which occupy the same level of abstraction and function as the ones that can be edited or created by the user. An example of a Rule in the engine could be the default character behaviour for reacting to being attacked. Examples of variables could be any of the properties which are fixed, such as a character's head size, the speed in which an item can be thrown and so on. Since these resources are not directly accessible by the user, it is not relevant to this analysis exactly how they are implemented in the system.

Software applications usually also involve "conceptual models", which are high-level abstractions used to describe the structure and logic of each one of the different elements and behaviours in the system. In a game engine such as MissionMaker's, conceptual models have a significant impact in the final product, defining how the 3D topology will be described and the structure of the rules system,

[40]As mentioned before, although this text often refers to aspects related to the execution of the code, the focus of this study is still on the creative (or design) aspect of procedural authorship.

[41]This is reflected in the fact that variables may exist by themselves, outside of rules, while rules usually depend on (and include) variables.

[42]Also, this analysis is slightly longer than the previous ones, of the procedural semiotic resources available to the user. This reflects the fact that, in MissionMaker, most of the resources exist exclusively inside the engine.

as well as how the player's perspective of the game world will be rendered on screen. Therefore, considered from the perspective of a procedural semiotic resource, these conceptual models provide a significant amount of meaning potential to the overall experience of these games.[43]

For example, consider the **representation model** used to describe the game world internally in MissionMaker. As mentioned before, this particular engine is heavily based on a metaphor of a 3D physical world, which is a common model in many modern digital games, such as First-Person Shooters, "platformers" and even the more experimental or artistic productions. The structure through which the world is represented inside the engine is described in the form of a conceptual model. This includes, for example, the information that objects have a certain shape and position in the 3D space, that they are submitted to certain laws of physics and so on.

Compare this to an alternative representation model, such as the one in the game Pac-Man (Namco, 1980), which still represents a somewhat physical space, however in a more abstract two-dimensional environment. Or compare it to a spreadsheet software application, which represents its content as numbers and tables, with no relation to space whatsoever (except for the user interface layer). The representational model does not refer to or contain information or data regarding the content itself - such as the 3D object data in MissionMaker, the graphics in Pac-Man or the numbers from a spreadsheet. It is solely a description of the structure and logic through which the actual content will be described, stored and presented. The same is true for all the other conceptual models.

Also, the actual contents that are informed by the conceptual models may be considered procedural semiotic resources themselves, even when they are static. As mentioned before when discussing variables, any static piece of data may be set up in a way that it can affect and contribute to the system's logic. An example of that in MissionMaker is the topology of the rooms, which may affect how characters behave and relate to each other. An example in which this does **not** happen is the topology of the characters, which in principle doesn't affect the actual behaviour of the system (in the simulation, all character types are considered in the same way, via their bounding boxes).

The **aesthetics** of a program can also be seen as defined by conceptual models, such as the rendering functions in a 3D engine used to display the environment, objects and characters on screen to the player. Considered as a semiotic resource, these functions may be used by the designer to express a certain aesthetics or even to represent particular ideas or concepts.

For example, the rendering functions in MissionMaker are relatively realistic, in the sense that the world is presented to the player in such a way as to simulate a first person perspective, following real world physics about how certain materials and textures react to light and so on. However, a rendering function can also be designed to provide a different aesthetic, such as enhancing the contrast of the image, in order to create a more dramatic experience, for example.

Although the functions that deal with the output of a system are inherently procedural, their end

[43]Some of the terms used to describe the high-level functions are specific to the examples presented here. Different systems, software and languages may utilize different terminology. Also, this particular definition of "conceptual models" may differ from that in other fields.

product is always static. For example, visual functions, such as the ones mentioned above, output images, while audio functions output wave content, and so on. This also means that these are the only semiotic resources that are directly accessed by the user or audience - any other resource depends on the output functions in order to be visualized or perceived.

The conceptual model for **playability** is also a particularly relevant one in MissionMaker, since games can only be fully realized through player interaction. In a MissionMaker game, the player can move the character through space using the directional keys, rotate its head using the mouse (or trackpad), jump, crouch and so on.

Considered as a procedural semiotic resource, playability may also have meaning potential in itself. For example, in a first-person game it is possible to express that the player's character is confused by inverting the key associations for movement - this makes the avatar go left when the right key is pressed, look down when the mouse is moved up and so on.[44]

Medium-Level Abstractions

In MissionMaker, medium-level abstractions could be the functions found one level beneath the actual software, which would be the programing language used to design it. Rather than discussing the specific programming language used to implement MissionMaker, here I am making reference to the level of abstraction in which it resides. In this case, these are the functions usually found in a programming language that operates in this level of abstraction.

For example, in this level there could be specific functions and procedures related to how this particular language deals with visual output, as well as input and data management. Programming languages also often provide their own implementations of certain functions such as sorting procedures, data structures or random number generators, which are commonly used by programmers as a means to introduce variation in a system. For instance, a randomizing function may be used to determine when and how the characters in a simulation use their idle behaviour animations. According to the language used to implement that simulation, a different algorithm could be used to calculate the random numbers, therefore affecting its outcome.

The actual contribution of this level of abstraction in the final product may be very subtle. However, in some cases the effect can be very significant, such as in applications that involve physics simulations - the functions used to calculate the different forces acting on the objects may drastically affect the resulting behaviour of the simulation.

Lower-level Abstractions

As presented previously, the lower-level abstractions are the functions that are closer to the fundamental logic of digital media, as well as to the hardware logic. They are more general in terms of function, and therefore less specific to any particular software. For example, while some high and medium level functions in the MissionMaker engine may exist only in this particular software (or only

[44]The "conceptual models" shown here, although considered to be high-level abstractions in the wider scope of this analysis, they tend to occupy different levels of abstraction in relation to themselves.

in 3D game engines), most of the lower-level abstractions are common to any game, software or language. This level includes functions, procedures and variables more closely related to the actual hardware behaviour. For example, the functions for actually sending visual output information to the computer screen, or the manipulation of data stored in the computer's memory.

On one hand, the fact that the functions in this level are not specific to MissionMaker (or to the programming language used to implement the software) means that they contribute less in defining the particularities of this application, in the sense of differentiating it from other productions in digital media. On the other hand, this also means that this level is the most influential, in the sense that it underlies every single software, programing language and operating system based on it.[45]

As an illustration, consider the fact that low-level functions represent a particular ideology, going back to the origins of the digital computers, which is defined by a systematic approach to data processing, as well as a direct connection to military interests. Although the meanings and implications related to these functions don't necessarily affect the use of modern computers in a direct sense, they are certainly present and contribute to the role of digital media in society.

It is outside of the scope of this text to discuss in greater depth how this affects the actual creative process of the game designer using MissionMaker, as well as the games themselves. However, recognizing this influence is relevant, in the sense that it directly relates to the choice of the creator to use this particular medium or language (or choice of MODE, as seen previously). A type of creative approach that directly addresses this issue is the practice of Circuit-Bending, for example, in which creators disturb or modify the actual systems, and even the circuits in the computers and machines used.

These are examples of just a few of the procedural semiotic resources that exist outside the reach of the game designer using MissionMaker. In all of these cases, the user can't influence these particular resources, their properties or behaviour, since they are contained in the software's engine. For example, the MissionMaker user can't choose a different rendering aesthetic, change how the player's input is interpreted by the system or define different functions for rendering 3D models.[46]

These resources may still express meaning, or at least represent a certain approach or ideology, which comes from the designers and programmers responsible for creating all of these lower-level functions.

These are all inherently **procedural** semiotic resources, in the sense that they exist solely in relation to their contribution to the system's logic. For example, consider the conceptual model for Rules in MissionMaker, which includes the components that make up this particular element, as well as how they are structured. This model only makes sense as a guide to how this particular function will be executed in the context of the program.

[45]The notion of levels - or layers - of influence was discussed previously in this text, in the presentation of the Framework and Methodology (p. 18).

[46]There is a certain degree of control over some of the lower-level functions and properties, however this is not common.

4.3 Composition

We have seen that the game designer using MissionMaker has access to two types of resources: **procedural** semiotic resources - the focus of this study -, which consists of the variables and rules (p. 29), and **static** semiotic resources, which may be textual, visual or audio content (p. 24). Now we will see how these resources are integrated into a single coherent system - the finished game -, which is done through the act of **composition**.

Previously, I presented the game design workflow in the MissionMaker editor, and how it emphasizes the spatial and physical aspects of the simulation. Because of that, in this particular software the act of composition usually begins by selecting and positioning the different elements in the game world, which is then followed by the definition of the rules, behaviours and variables (p. 29).

The diagram below (fig. 14) represents how the semiotic resources are organized in MissionMaker, and how they may be accessed by the designer. It also includes the resources found in the game's engine and the software itself, which are not directly accessible to the user. Static semiotic resources are included as well, since they are an integral part of the game design in this software.[47]

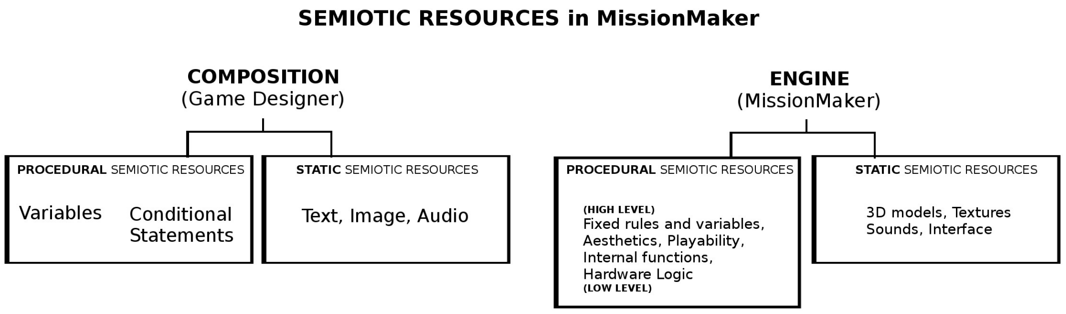

Figure 14: Diagram for the Semiotic Resources in MissionMaker

Procedural Authorship

Composition in digital media is analogous to that in other languages, such as visual design, music, cinema and so on, in the sense that it involves organizing smaller elements in order to form a larger structure, which usually expresses or represents an idea or meaning that each element couldn't represent by themselves individually (or if grouped in a non-structured way).

In this study, composition is considered as a semiotic act involving the selection, combination and arrangement of semiotic resources in order to produce a particular message or meaning. In that

[47]As mentioned previously, static semiotic resources are any textual, visual or audio content that does not participate in the logic of the system - as such, they don't contribute to the procedural meaning potential, although they may provide static meaning potential (p. 24). In the diagram presented in this page, the static semiotic resources listed as part of the Engine refer to the content that comes as part of the software, and that can't be modified by the user (character, object and location models and textures, default sound effects and so on). The diagram also references topics introduced previously in this section.

sense, the process is analogous to the act of combining words to form sentences. Applying this analogy to the creative process in MissionMaker, the "grammar" through which these sentences can be written would be the particular structures and relations through which the designer may compose the semiotic resources, and the "text" would be the final game (or a particular element or subset of it).[48]

As in other languages and tools, the grammar of MissionMaker is what defines the type of creative process and output associated to it. It is the grammar which provides the particular constraints (via rules and structures) through which the act of composition may take place. These constraints are necessary, since they define the characteristics of this particular language.

Besides the constraints that define MissionMaker as a game creation tool (instead of a word processor, or a low-level language, for example), there are also the constraints that make this particular software more accessible, as mentioned previously (p. 33).

"Composition" is being considered here as the act itself, instead of the elements involved in it, or its result. For example, in MissionMaker choosing the position of a certain character in a room is an act of composition. This could also be described as the modification of the variables that store this particular object's 3D position in the world, for example. However, from the perspective of the creative process, this is still an act of composition.

There is, however, a fundamental difference that distinguishes composition in digital media from that in other languages: the fact that it involves **procedural** semiotic resources, which allows for a unique type of expressiveness - or meaning potential -, as explained previously in this text (p. 6).

Additionally, in digital media the act of composition itself can also happen procedurally. That is, it may exist in the form of an algorithm or procedure, which describes the structure or logic of the composition, allowing it to be applied or executed dynamically to any given set of elements (considering that they fulfill the format and quantities required by the algorithm).

In MissionMaker, this is the case of Rules, for example - their structure represents a particular type of composition involving certain types of smaller elements, or units. This doesn't change in any way the nature of the act of composition itself, its role in the creative process or its expressive qualities - it is just shifted, from the mind of the creator to an algorithm. In other words, in this case, the composition becomes a "metacomposition".

We have seen that semiotic resources allows the expression and representation of meaning potential. The act of composition can also be expressive, however, as mentioned before, it operates in a different semiotic level. Composition produces meaning potential via the selection and combination of existing semiotic resources (some of which could be, as mentioned before, “metacompositions").

[48]Compared to other game creation environments which allow the designer access to more low-level functions, MissionMaker offers a relatively limited grammar. As mentioned before, this is a design choice, in order to make the process more accessible. However, most of the concepts discussed here apply to digital media in general.

The type of meaning potential represented via a particular act of composition - if it is going to be static or procedural - is directly dependent on the elements it involves, how they are organized and related to each other.

For example, if the placing of a certain object into the game world does not affect any other object, character or rule, and if that object is itself static as well, then this particular act of composition has no **procedural** meaning potential. An example of that would be an object that cannot be picked up or interacted with in any form by the player or any other character.

The placement of this object could still produce **static** meaning potential, if the choice of having that particular element in that particular context had some kind of narrative or expressive relevance. For example, the designer could place a single nail in a wall, in order to suggest that a picture frame has been removed from it. In this example, what is being considered as expressive or meaningful is not the semiotic resource itself (in this case, the nail), but how it was **composed** within the system.

4.4 Comparing MODES: Shared Semiotic Resources

We saw how a game designer selects, creates and composes procedural semiotic resources in MissionMaker to express ideas and represent meanings. I also mentioned that, in order to further investigate the particularities of this process, it is useful to compare it to other languages and mediums - or MODES - which are more established and mature. This will be elaborated further in the following pages, by identifying and comparing shared semiotic resources between the game design process in MissionMaker and the other creative activities from the Playing Beowulf pilot workshop.

This analysis will be based on the consideration of Procedurality as a MODE, as well as on the significance behind choice of MODE, mentioned previously in this text, in the Framework and Methodology section (p. 16).

What follows is an initial investigation on the parallels between digital game design and other languages and mediums with similar expressive strategies and creative process. As we will see, some of these intersections are more apparent than others, and some have a greater influence or effect in the actual productions, while in others this is more subtle, or even non-existent. The intention is that the ideas presented here may be the starting point for further elaborations on this subject.

MODE as Groups of Semiotic Resources

The argument that I will elaborate on here is that the different languages, practices, artforms or MODES can be described via the semiotic resources on which their respective creative processes are based on, as well as on their particular grammar for composing these resources.

As seen before, semiotic resources may exist in a variety of different levels of abstraction, according to the particularities of each MODE, as well as according to the perspective and approach from each author or creator.

The following diagram (fig. 15) illustrates this, using as reference the languages explored in the workshop, including the game design activity. The dots represent the different existing semiotic resources, and each grouping represents the respective languages, such as drawing, film or drama - intersections between the groupings illustrate the shared resources. Although the dots may represent static semiotic resources as well, the focus here will be on the procedural ones, since this is the main topic of this study.

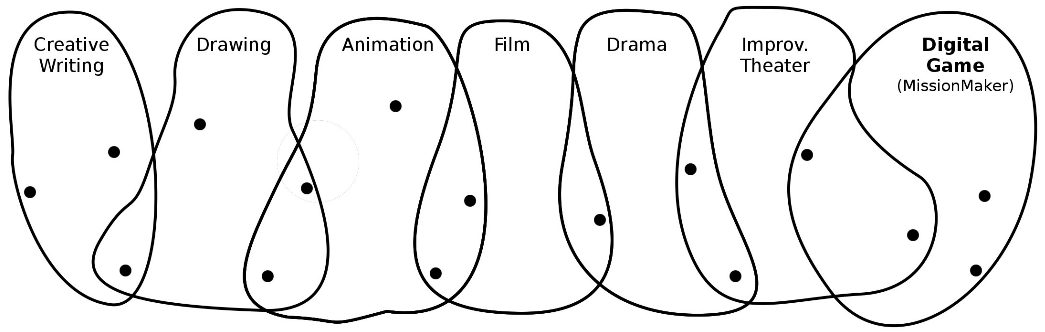

Figure 15: Semiotic Resources: groupings by language (or MODE)

This diagram is a greatly simplified representation of these elements and relationships. For example, regarding the semiotic resources (the dots), the diagram doesn't differentiate among the different types (static or procedural), neither among different levels of abstraction. Also, the number of dots represented in the diagram does not reflect the actual quantity of resources involved in each language, or the intersections between them. Not only is the actual number of resources much higher than what is represented in the diagram, in principle they should be unlimited, specially considering that resources can exist in different levels of abstraction.

These groupings are not meant to be rigid. Although there are certain sets of resources which are usually associated with each language, the groups are dynamic and flexible (also, groups different than the ones presented here could be defined as well). Therefore, the actual groupings may vary according to different perspectives, as well as to the particular creator or artist.

The intersections between the groups are also represented in the diagram in a very simplified manner, since each MODE usually shares semiotic resources with many of the other MODES (instead of only one or two, as shown in the diagram). For example, while color is a semiotic resource shared by many languages, texture is present in fewer MODES. The abstraction of functions (such as the ones mentioned before) in a programming language are semiotic resources shared with even fewer MODES.

Regarding levels of abstraction, on one hand, resources in lower levels tend to be shared between more MODES, since their usage tends to be less specific in terms of application and thematic - this is the case of fundamental properties, such as color and texture, for example. On the other hand, certain higher-level abstractions may also be shared among many MODES, in the cases in which they are not specific to particular medium or support - such as the idea of concept models, mentioned before (which can be applied outside of digital media as well, for example).

Shared Semiotic Resources

During the Playing Beowulf workshop the participants were exposed to a variety of languages, and encouraged to act creatively using them, in a very short time span. Often the activities required two or more of these artforms to be combined or integrated in some way. This allowed the participants to gain a very unique perspective on each one of these approaches, as well as to the general concepts of language and creative expression.

In my conversations with them, many had opinions regarding the particularities and similarities between the different mediums and languages, both traditional and digital. In the following pages I will present some of these discussions.

As mentioned before, the kind of comparisons suggested here are useful for illustrating and understanding the concept of semiotic resources, as well as the role of particular semiotic resources in each language.

Since most of the participants are student teachers from the English and Drama course, and almost all of them teach Creative Writing or some other closely related subject, the medium of literature and verbal language was one of the most common examples used in comparing game design (and digital media) with more traditional approaches.[49]

Niamh Hickey, when comparing the creative process in MissionMaker with the activity of creative writing, pointed out that game design allows for much more immediate results. She argued that a writer always starts a new work from scratch, not only because the process usually begins literally with a blank piece of paper, but also because the whole world logic needs to be described and defined, as well as the structure and style of the narrative. In a creative environment such as MissionMaker, on the other hand, Hickey believes you get most of that by default.[50]

What I found, however, is that in both cases the same process happens, only in different ways. The writer does not actually start his work "from scratch", at least not in the sense of the comparison presented above, since a great part of the work already happens even before picking up the pencil (or computer). Also, all of the "default" elements mentioned in regard to MissionMaker had to be created by someone - in this case, the designers and programmers of the software.

Therefore, the difference pointed out by Hickey exists only from the perspective of the game designer using MissionMaker (or a similar type of software). From the point of view of the MODES themselves, however - game design and creative writing -, both share the same "world-building" process, which can be described as a particular set of semiotic resources.

The designers and programmers of MissionMaker (the software itself) defined the world's logic, and many of the rules, behaviours and even assets involved in it. Similarly, a writer is responsible to decide how the world inside a particular story will work. While the low-level semiotic resources used in each case may certainly differ (the programmer uses functions and variables, while the writer uses

[49]Another medium that was mentioned by many of the participants was Cinema, a popular artform, which makes it easier to relate, exemplify and find common references and expressive strategies to compare to digital media.

[50]This can be related to the idea of the game engine, discussed earlier in this text (p. 38).

sentences and words), resources in higher levels of abstractions are the same. This includes, for example the description of characters and locations, as well as the definition of relationships and rules through which the systems will work.[51]

Harriet Piercy mentioned the similarities between game design and the improvisational drama exercises done in the workshop. In that exercise, the participants didn't receive a script with a detailed description of their lines and actions in each particular scene. They received more general instructions, such as to react to a certain situation, or to act in a way that a certain character would. Although in a way these are also scripts, they don't represent linear static instructions. They represent behaviours and processes instead.

That is, in that exercise the instructor was in a way "programming" the actors, in a process similar to that of a game designer programming characters in their games. Again, the shared resources in this case are the ones in a higher level of abstraction - after all, people can't literally be "programmed", at least not in the same way as computers.

Related to this topic, Niamh pointed out that in many of the activities from the first day of the workshop the participants were placed as active characters in the story world. This helped them in getting into the mindset necessary for designing a game, since this is a medium usually centered around the notion of an active central character controlled by the player, in which his actions and decisions define or influence the outcome of the experience. This suggests the existence of shared semiotic resources related to this particular aspect of the experience.

The idea of shared semiotic resources between MODES can also be used to illustrate an observation by one of the participants of the workshop, Laura Scott. She suggests one possible model for integrating game design and creative writing in an educational environment, by having both activities happening almost simultaneously, allowing constant feedback and insights between the two approaches.

For example, a student could create a game, then write a script (similar to a theater play or movie) that another student could use to interact with the game - what Laura proposes can also be related to the concept of a videogame walkthrough (BURN In CARR et al, 2014). This process could then inform a new round of game design, script writing and interaction, gradually evolving the narrative associated with this particular experience. By requiring these two approaches to be pursued practically at the same time and in the same context, Laura believes this could cause these drastically distinct languages to interact in the student's minds in ways that they wouldn't otherwise. This could bring new ideas and ways to use these languages, both individually and as a cohesive unit.

In this exercise proposed by Laura, there is always a bridge between the different mediums, which mostly consists of their shared semiotic resources. For example, throughout the entire process, the characters, locations and relationships are the same (except when the game designer chooses to

[51]In both cases, the work may not even involve a "world" in a strict sense, or even a "story", for that matter. Also, the comparison proposed here is in relation to elements sharing the same expressive purpose in the respective languages. So, although games may also contain "words" and "sentences", they don't participate in the same type of meaning expressed via the procedural elements, for example.

update or adjust them - which is the very purpose of the exercise), while resources specific to each language (such as the linear structure of verbal language, in contrast to the rule-based logic of the game) vary.

As mentioned before, this is an initial investigation on comparing MODES considering the shared semiotic resources between them. There are many other aspects and elements that can be considered in further elaborations, such as comparing the role of **composition** among different MODES, as well as how their overall creative approach relates to each other.

The languages and mediums used in the comparisons allow for interesting insights and discussions, mainly because of the associations resulted from the activities in the workshop. However, these are not necessarily the languages that could provide the most significant comparisons to the game design activity. Therefore, this investigation could benefit from further analysis considering comparisons with languages and practices more closely related to the creative process in digital media, such as the aforementioned traditional forms of Conceptual and Kinetic Art, which shares many elements and expressive strategies from the procedural approach (p. 14).

5 Conclusion and Further Research

In this text I hope to have provided a useful perspective on how digital media can be used to produce meaning via procedural expressive strategies. This was illustrated by some case studies, such as Andrew Smith's game, in which he used particular behaviours for the representation of Grendel's mother as a peaceful but strong character, or Tracey Matthews' game, which communicated the relationship of loyalty between the warriors and Beowulf via the definition of simple rules.

Despite their simplicity, these examples demonstrate the potential of an approach which is unique to this medium, fundamentally distinct from any previous traditional language or form, requiring a different way of thinking about representation and expression, not only from the perspective of the creators, but also of researchers and critics, as well as the audience.

I should reiterate that the analysis of the MissionMaker software done in this study is very specific, with focus on the procedural semiotic resources, as opposed to static elements such as the use of images, text, sound and music. As mentioned before, these static elements are a fundamental part of most of the games analyzed in this study. Partly this is because of the fact that, for most of the participants of the workshop, this was their first experience creating games. However, this is also related to the broader notion that the expressive potential of procedural authorship is still relatively unexplored, as discussed previously in this text.

This shouldn't be interpreted as a criticism to the games or their expressive qualities, neither to the software itself. First because, as mentioned before, the limited functionalities available in MissionMaker are designed in order to allow for a more accessible experience, targeted for inexperienced game designers and computer users. Second, because the fact that procedural semiotic resources are not the predominant mode of expression in these games does not mean that there is no meaning potential being communicated via the static resources.

A significant finding that resulted from this study is that, despite the discrete and systematic nature of digital media, the fact that it acts as a metamedium means that there are no predefined notations regarding the language itself, through which creators and designers may express themselves and create meaning. In other words, although the code that underlies the operations of a digital computer is strict, and needs to follow the guidelines and rules of particular programming languages or hardware devices, the actual systems that can be created by them have no preestablished grammars.

The creative act of writing code shares many similarities to the structure of languages with well defined notation, such as verbal language or music. However, the actual moment of execution, or the possible outputs, of a computer program have more similarities to less structured languages such as painting, dance, conceptual art, drama and improvisational practices.

The importance of examining this particular aspect of digital media is to contribute to a better understanding of the expressive potential of this medium and of how it relates to (and, more importantly, distinguishes itself from) other traditional forms.

Initiatives to bring programming and game design to schools in an interdisciplinary context, such as Scratch (MIT) and MissionMaker, tend to focus on the more technical aspects and applications, instead of on creativity and expression. Programming tends to be taught in the context of Computer Science, as opposed to Art or Cultural Studies. Using an analogy between programming and verbal language, it is as if there were many classes for learning the grammar, but few for "creative writing".[52]

Note that the focus of analysis here is the use of game design and procedurality for expressive and representational purposes. In that sense, it differs from the topic of gamification in education, which deals with the use of game design for learning other subjects, including programming.

Regardless, procedural literacy is an issue even among experienced programmers and creators in this medium. As mentioned before, although there is no "wrong" way of using or enjoying digital media, the unique expressive potential of this medium is still relatively unexplored and misunderstood, both by creators and the audience.

This study is part of a broader PhD research I am currently developing, which analyzes procedural authorship in a more general sense, considering not only digital games but also other types of productions based on procedural strategies of expression. Through this research I hope to contribute to the understanding of digital media as an expressive medium, by providing concepts and theoretical tools for investigating how artists and creators use computers to produce a type of meaning potential that is unique, different than in any other traditional medium or language.[53]

[52] I intentionally avoided the term "creative coding" here, since it is often used in reference to a creative approach to the act of coding itself, instead of to the creative uses or applications of coding.

[53] This research follows the theories developed in my previous masters dissertation (FERREIRA, 2011). More information at my research blog: `www.7luas.com.br/research`

References

Note: all online references were accessed and verified at July 2015.

ALBERRO, Alexander; STIMSON, Blake (Ed.). Conceptual Art: a Critical Anthology. Massachusetts: The MIT Press, 1999.

BOGOST, Ian. Unit Operations: an approach to videogame criticism. MIT Press, 2006.

BURN, Andrew. 'The kineikonic mode: towards a multimodal approach to moving-image media'. in The Routledge handbook of multimodal analysis. 2nd ed. edn, Routledge. 2013 pp. 373-383.

COHEN, Harold. Parallel to perception: Some notes on the problem of machine-generated art. Computer Studies, v. 4, p. 124-133, 1973.

CRAWFORD, Chris. Process intensity. Journal of Computer Game Design 1(5), 1987.

FERREIRA, Daniel Peixoto. Poéticas procedurais: um olhar sobre o pensamento artístico e a expressividade do meio digital. Masters dissertation (portuguese). ECA/USP. São Paulo, 2011.

________. Beowulf Workshop: A MissionMaker Case Study. Nov. 2014a.
URL: `darecollaborative.net/2015/03/11/playing-beowulf-gaming-the-library`

________. Beowulf Workshop: MissionMaker Conversations. Dec. 2014b.
URL: `darecollaborative.net/2015/03/11/playing-beowulf-gaming-the-library`

JEWITT, Carey (Ed.). The Routledge Handbook of Multimodal Analysis. Routledge. 2014.

KAY, Alan. Computer Software. Scientific American. vol. 251. num. 3. pp. 52-59, September, 1984.

LEEUWEN, Theo Van. Introducing Social Semiotics. Psychology Press, 2005

MANOVICH, Lev. The Language of New Media. Massachusetts: The MIT Press, 2001.

MATEAS, Michael; STERN, Andrew. Procedural authorship: A case-study of the interactive drama façade. In: Digital Arts and Culture: Digital Experience: Design, Aesthetics, Practice (DAC 2005) Proceedings. Copenhagen, Denmark, 2005.

________. Semiotic considerations in an artificial intelligence-based art practice. Dichtung Digital: Journal on Digital Aesthetics, n. 29, 2003a. URL: `www.dichtung-digital.de/2003/issue/3/Mateas.htm`

________. Expressive AI: Games and artificial intelligence. In: Level Up: Digital Games Research Conference Proceedings. Utrecht, Netherlands, Nov. 2003b.

MURRAY, Janet Horowitz. Hamlet on the Holodeck: The Future of Narrative in Cyberspace. Simon and Schuster, 1997.

REAS, Casey. Interview at Electric Objects. Feb. 2015.
URL: `zine.electricobjects.com/interviews/casey-reas`

SICART, Miguel. Against Procedurality. Game Studies. Vol. 11, Issue 3. December 2011.
URL: `gamestudies.org/1103/articles/sicart_ap`

TONSKI, Jacob. Prix Ars Electronica 2014. Artist's statement. Digital Document. 2014.
URL: `prix2014.aec.at/prixwinner/13363/`

WARDRIP-FRUIN, Noah. Expressive Processing: On Process-Intensive Literature and Digital Media. PhD thesis, Special Graduate Studies, Brown University, Providence, Rhode Island. May, 2006.

Software and Technology

Actionscript (Macromedia Flash)

MissionMaker (London Knowledge Lab, UCL Institute of Education)

Processing (Ben Fry and Casey Reas, 2001)

Scratch (MIT)

Digital Productions, Videogames and Artwork

Aaron (Harold Cohen, 1973-)

Balance From Within (Jacob Tonski, 2012)

Cutthroat Capitalism (WIRED, 2009)

Façade (Michael Mateas and Andrew Stern, 2005)

Far (Random Dance, 2011)

For All Seasons (Andreas Müller, 2004)

Hana (Andreas Müller, 2008)

Oregon Trail, The (MECC, 1971)

Pac-Man (Namco, 1980)

Sand Installation: Ryo-Anji (Jean-Pierre Hébert, 2011)

Spelunky (Derek Yu, 2008)

Super Mario Bros. (Nintendo, 1985)

Traditional Productions and Artwork

Beowulf (modern English translation by Frances B. Grummere).
URL: `www.poetryfoundation.org/poem/180445`

Cent mille milliards de poèmes (Raymond Queneau, 1961)

Choose Your Own Adventure (series, Bantam Books).

One Hole in the Ground Approximately 1' x 1' x 1' /
One Gallon Water Based White Paint Poured into this Hole (Lawrence Weiner, 1968)

Terminal, The (Steven Spielberg, 2004)

Theater of the Oppressed (Augusto Boal,1960s-)

Index

The following is a brief index to some relevant terms and concepts in this text (this is not a list of all instances of these terms, only their main occurrences).

[End of Document]