UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Dynamic theme-based narrative systems

Basart i Bosch, Ferran-Roger

Director: Pons López, Juan José

Degree in Videogame Design and Development 2021-22

Centre de la Imatge i la Tecnologia Multimèdia (CITM)

Table of contents

Abstract

The advent of videogames, and the new forms of expressions they offered, sprouted the possibility of presenting narratives in ways that could capitalize on unique qualities of the media, most notably the agency found in their interactive nature.

In spite of many people in the game studies' field interested in how far said novelty could bring narrative experiences, most approached the creation of narrative systems from a structural approach (especially the classical Aristotelian one), and concurrently, with a bottom-up (characters defining a world) or top-down (world defining characters) perspective.

While those more mainstream takes have been greatly progressing what interactive digital narrative can be, this research intended to take a bit of a detour, proposing a functionally similar system that emphasized thematic coherence and responsiveness above all else. Once the theoretical formulation was done, taking into consideration previously similar or tangential systems, a prototype would be developed to make a first step towards validating the proposal, and contribute to building a better understanding of the field's possibilities.

Conclusions

Whilst thematic construction retains its relevance in the story-telling process of the videogame media, the parametrization of such abstract concepts can hardly be replicated in the same way that logical sequences can. The most effective method to imbue the narrative with thematic depth, then, becomes to manually define how and where to include it, thus making it a layer outside of system automation, with the authorial burden it entails, but also the greater, necessary control that the task requires.

Keywords

Interactive narrative, emergent narrative, drama manager, themes, planning system, content authoring, videogames.

Acknowledgements

This project would have most likely yielded lesser results if not for the references provided by my tutor Juan José Pons López, as well as the material provided by Carles Homs Puchal in both structuring and formatting.

List of figures

0. Preface

Before introducing the research properly, a run-through of personal appreciation to narrative would be appropriate to contextualize both this project's origin, foundation and development, even if the topic of themes will be further discussed later on.

From all the aspects of narrative (here understood as the curated discourse of a series of events), themes are one of the, if not the most, abstract ones. Despite the abundant use of the word to refer to the esthetics of an element (for example, a space theme or a winter theme), the usage of theming throughout this research will exclusively reference the non-verbal descriptive attributes of a narrative body, or put in simpler terms, what a narrative is essentially about beyond the moment to moment occurrences of the story. Themes do not merely refer to specific concepts, but to sensations and feelings too; as long as they are the (or one of the) fundaments of the narrative sensorial core.

H. Porter Abott wrote "identifying themes and motifs can help enormously in establishing what a work is about and where its focus lies" (Abott, 2008). This revealing capability, able to recontextualize obtuse human interactions, convoluted plot lines, demential worlds, idyllic scenarios…, into clear and personal ideas; able to achieve a purely emotional communication between author and audience, the expression of universal human experiences in universally real terms, without the interference of a learned language that enables the understanding of the message, is what drove this research to its completion.

1. Introduction

We would like to start with a disclaimer, as this project is an individual effort to make a C++ coded system that manages multiple aspects of narrative from a theme-based perspective, oriented to being used in a videogame context.

While many related concepts will be explained as the report goes on, valuable sources to contextualize the theoretical space in which discussions will take place are *Avatars of Story* (Ryan M.-L. , 2006) and *Interactive Digital Narrative: History, Theory and Practice* (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017), and its advised (albeit not necessary) that less familiar readers give them a try before venturing into the following pages.

1.1. Problem definition and secondary problems

A very centric point of interest in the interactive narrative field is the potential of videogames to force the player to take compromising decisions, which is what this research believes to be the core of the media's artistic unique capabilities: being able to inform players about specific human experiences by forcing them into the position of actors instead of spectators. In order to realize stories that enable such enriching contexts, we have to offer emotionally meaningful situations to the player, that make the choices significant, but the creation and polishing of a setting that delivers a proper experience usually runs into a series of contrivances intrinsic to the media's properties.

To begin with, there is a conflict between authorship and player expectations, in so much that every decision has a limited amount of responses that the developers can account for. If what the player expected to happen contradicts the results, or has not been contemplated, frustration will emerge, and based on how crucial the decision was, many choices in the future might feel invalidated because they are a consequence of a disconnection between player and game. This problem emerged early on in the production of videogames, as there can only be so many routes to explore through fully manually authorized content.

Another factor to consider is to keep the game's presented ideas clean after introducing a regulating system: take as an example an NPC whose beloved brother we killed, but with which we have a friendly connection, because the sum of the parameters that define relationships

indicate a positive result. Independently of whether this would be realistic or not, we might be incorporating themes of forgiveness in a story that did not contemplate them as a result of defining what a system's interactions can result into; in other words, it is not the designer who is taking the decisions, but, in this case, a blind system balancing favor points.

Games can also easily suffer from utilitarian attitudes when mixed with meaningful choices: we do not want the player to choose by what will yield the best or most optimal results, but by their personal believes and unknowns. Opposite to this, if the choice does not have meaningful consequences, as to avoid utilitarianism, it is much harder to consider it relevant. Realistically, this dichotomy only exists for a certain type of player, since the dismissal of the narrative in favor of gameplay is a playstyle not uncommon in the vast pool of people who play games, but the real issue is not that narrative is ignored, it is the extent to which authors condition utilitarian attitudes when players that do care about the story are faced with the uncertainty of a choice's results. This is deeply related with theme pollution, as this kind of decisions can be used for meaningful storytelling, but more often than not, creep into narratives that should not concern themselves with utilitarian discussions.

Another common problem to making meaningful worlds is in how the application of mechanics affects the narrative. The clearest example is that of a character with a sword that can be swung anywhere, but which cannot be used to kill NPCs. The designer's answer is that it is too expensive to plan for a believable reaction to killing every NPC, or that the narrative cannot afford losing certain characters, but no diegetic reasoning is usually given, accepting it as a "limitation" or "convention" of the media instead of a problem to be solved. And yet, it is a disruptive element, or at least an anti-immersive one.

Finally, meaningful choices are defined by their consequences. If the player has any way to omit, revert or correct such consequences, even if it is not always undesirable, it will easily devalue the relevance of the choice. This includes the abuse of save – load points, but also systems like the aforementioned relationships-by-points systems, where one can farm a certain action to unrealistically undo any and all committed transgression. The other side of the coin is that, by taking away the opportunity to revert a decision, we end up accepting misunderstandings and errors as inputted choices, which will inevitably frustrate the player if they occur.

While many of the previous problems apply to any narrative choice that we implement, this research is especially interested in considering this problematic from the most complex context

of freedom-centric games, where being able to interact with the narrative in "unintended" ways is part of the core experience. Examples of it are the *Dark Souls* saga (especially the first installment) or the *Fallout* games; note how we do not necessarily refer to open world or sandboxes, but to games that allow narrative interaction through performed, and not only selected, gameplay. This distinction was deemed necessary to clarify that we will not be limiting our approach to text-based games, as many similar researches do.

1.2. Research question and secondary questions

The main research question is about how could we design a system in a narrative-heavy game (both for mechanic and static choices) that allows a multifaceted, meaningful player response, reciprocated accordingly to the specificity of the input given, through the use of themes.

In order to satisfactorily solve this question, the following must also be answered to a certain extent:

- How can we account for an adequate consequence for every action the player takes without skyrocketing the production cost?
- How can we orientate the player towards understanding the consequences of their actions without spoiling the actual results?
- How can we implement our regulating system while keeping the creation's themes, that is, empowering the designers through the system and not vice versa?
- How does the system handle reward display to be satisfying while not conditioning the motives behind player agency?
- How does the system account for mechanic-based actions that may have narrative consequences?
- How does the system handle choice permanence and going back on a decision?
- How does the system consider allegiance thresholds and points of no return?

1.3. Main and secondary objectives

According to the defined main question, the core objective will be to design a system that allows for an appropriate response to a wide variety of player actions, paired with providing a code

structure that can serve as the base or prototype of such system. Following this first premise, we can define minor goals that would be desirable.

1.3.1. Input flexibility

The system must remain organic and unspecific, monitoring any desired key actions performed by the player or any other entity, and not limiting ourselves to specific types of events: ultimately, it should be serviceable to any designer in mostly any context that involves narrative management.

1.3.2. Input complexity

The system must allow the registered actions to be complex, going beyond simple mechanical input, and should be extendable to complex chains of distinct actions, which may or may not be influenced by other complex conditions (location, relation with previous events, order of areas explored…).

1.3.3. Reaction persistence

The system should be able to account for a wide variety of self-individual states that condition future input, that is, we not only take notice from the player's actions, but the state of the system (derived from previous decisions) affects the result of the incoming action, as to account for consequences throughout time. This means at some point we must be able to create internal rules that define characters' / factions' allegiance and points of no return, both positive and negatively.

1.3.4. Reaction interconnectivity

The system should be capable of describing to narrative state of an entity across many systems (NPC managers, event handlers, quest managers…). It would be desirable if it could express undetermined states: we could have a situation where the player can take three different routes out of five, and we may need to know which of those are available to determine the reaction of a certain NPC. The system, therefore, has to be able to handle and describe such structures

efficiently, and at some point should be able to create internal rules that define characters' / factions' responses towards other entities as a result of player or other's entities' actions.

1.3.5. Ego layers

The system must provide some level of abstraction between one or multiple individual character features, and one or more faction-based features, as to more easily organize and apply clusters of instructions that distinguish the "rational process" of different characters. It is to be noted that some instructions may be contradictory, but valid for one same entity; such incongruence should be solvable to allow for more flexible character individuality, be it by prioritizing the self or the group. Coincidentally, both an entity's said priority and allegiance should be alterable.

1.3.6. Output flexibility & complexity

In the same way that player response has to be varied, we must be able to communicate a series of more or less complex actions to an entity based on the output, and we must account for a flexible range in each of said actions, which should go beyond single mechanic outputs.

1.3.7. User accessibility

Since the system is devised as a tool that aids other developers in their narrative endeavors, it is an imperative that it has a clear structure, with clean code, easily portable and usable in multiple narrative and ludic contexts.

1.4. Glossary

1.4.1. Vocabulary

Bug: malfunctioning of an intended mechanic or dynamic due to imperfections in its implementation.

Choose your own adventure game: game genre that comprises those where player choice determines the progression of the story meaningfully, traditionally through very limited interaction among many text, connecting story nodes through basic player choice.

Diegesis: the reality comprised by everything that exists in a fiction world; the fictional world in itself.

Exploit: abuse of a characteristic of a system or element of a game to the player's advantage, be it overseen or unplanned by the developers.

Game Master: members of a role-playing game that are in charge of managing the state of play, in contrast to those who engage the diegesis by embodying a character in it.

Glitch: malfunctioning of a game that results in an unintended mechanic or dynamic due to conflictive code.

Godmode: debugging mode that intentionally allows bending the rules of the game to test its features.

Hypertext: in videogames, it refers to text-based games where only fractions of text are available at a time, the traversal of which is done through hyperlinks.

Ludonarrative dissonance: situation in which the ludic (mechanics, the actions) side of the game reflects a partially or totally opposed reality from the one the narrative is presenting.

Magic circle: state of being in which the individual perceives their reality as that of the context in which they are focusing; closely related to concepts like immersion and suspension of disbelief.

McGuffin: narrative foil that motivates conflict, but which remains irrelevant to its progress, usually serving as a shared final goal between multiple parties.

Open world games: those where the focus is the exploration and discovery of vast and interesting areas that are not separated by artificial game barriers.

Permadeath: mechanic where an entity's final lose condition marks the end of possible interactions with the player permanently, usually presented as a character death, hence the name.

Playerbase: group of people that has played a game, usually referring to those that have developed a minimal attachment to it.

Playstyle: group of distinct tendencies and behaviors that a player exhibits normally when performing in a game. In fighting games, for example, while crouching attacks may be commonly used by everyone as a universal mechanic, specific, repeated usages of the move can be described as part of someone's playstyle.

Playthrough: the total of game sessions that conform the player experience within the game from the beginning to the end (here to be understood as the moment in which the credits roll).

Replayability: the property of a game that provides engagement despite (or because) it is played repeatedly past the point of completion.

Sandbox games: those where there is a focus on experimenting with the elements that compose the game's world in order to create satisfying, usually creative, results and discover unexpected interactions.

Speedrunning: the practice of trying to complete a game as fast as possible, usually divided in glitch runs, which allow the use of glitches and non-intended exploits to complete the game; non-glitch runs, where the game is completed using only the developers' "intended" mechanics; and challenge runs, where the player aims to complete a goal while abiding for external, self-imposed rules.

Step: to iterate. In videogames, the term is usually used to refer to "stepping the world", which means running all the interactions predefined in a world by a specified amount of units (usually of time) in order to change its state. The expression, however, can be used for anything that self-regulates state.

1.4.2. Acronyms

AI: artificial intelligence.

CYOA: choose your own adventure (games).

HTN: hierarchical task network.

NPC: non-playable character.

RPG: role-playing game.

2. Methodology

2.1. Project definition

This project was conceptualized as an applied research, that is, it focused on answering a specific question, that being the possibilities of implementing a dynamically controlled narrative, based on thematic interactions. The development process was envisioned as an incremental delivery of content divided in four differentiated stages that would revolve both around user-centric design and expressionistic goals:

2.1.1. Planning

The first stage was to organize all the tasks to successfully understand what time window we had in order to do the necessary research, and how to distribute the workload among said period.

Since the development process was based on iterative methodologies, the planning was a stage repeated multiple times: first was the planning for the theoretical phase; then at the end of February 2022, to prepare and fulfill the first assignment's conditions (scheduled for the 25th March), and to organize the project's implementation. After the first revision, the second assignment (13th May) was taken into account, and after it, the final planning to ensure the completion of the system's prototype, as well as the final version of the project's report.

Depending on the stage of the project, different methods were used, from agile methodologies with online tool, to simple notebook lists that were kept updated on a weekly basis.

2.1.2. Information gathering

The second stage consisted of building a solid foundation through exploring the subject's state of art, as well as providing a handful theoretic framework to deepen and expand the understanding of any relevant element to the conversation. To this effect, we consulted plenty of material from many media (books, articles, videos…) and engaged with, and analyzed, various cases of study that were deemed useful for reference.

2.1.3. Technical demo

A substantial part of the project was spent on figuring how to apply the gathered knowledge and noted observations into a practical, usable basis that fulfilled the previously mentioned needs.

With the collected knowledge after the "information gathering" phase, it was defined that the solution provided to the problem would not only reference previous works, but that it would have to adapt what had been previously stablished by the community, since other similar systems had already been attempted. The one distinct factor, thus the one that would require further design of unique solutions, was the implementation approach centered around themes.

The technical demo was done in raw C++ to allow for easy portability and higher efficiency, and was designed on two fronts: the demo itself and the tool. The demo included a simple world set-up that serves as a showcase of all of the tool's functionalities, while the development process focus' was put on the tool itself, and its usability paired with clean code. As a consequence, the demo only took as little as needed to exemplify the tool's capabilities, avoiding unnecessary waste like graphics for world representation.

2.1.4. Closure

At the end of the process, some time was saved to polish this report and the technical demo to a satisfying end. As it was the case with the planning, this was also an iterative routine, meant to gather feedback and re-contextualize the project's state at the end of each assignment, as to better plan the next milestone.

2.2. Scheduling

To have a better grasp on the research's state, a Gantt table was devised, with the considerations that work began on the 4th October, with the initial planning being completed earlier than the 10th of that same month, the self-imposed limit date to finish preparations. Concurrently, the established limit day for the research was the 30th June.

Under tutoring guidance, and knowing that there would be more personal burden during the first semester, the one dedicated to the documentation, it was decided to divide time between

the theoretic and practical parts slightly benefiting the prior, thus ending the first half on late February. With enough margin to prepare the first assignment (25th March), multiple fields were considered necessary to explain / explore:

- Artistic notions and how it affects the implementation
- Theming in narrative, and its videogame's specifics
- Agency, emergent narrative and interaction
- Examples and alternatives to emergent, dynamic narrative systems

At a micro-level, a minimum of four daily hours were expected for the first half of the project, while the second half would allow for a more generous minimum of six daily hours. This approximation did not account for time spent in university studies, but it also did not account for personal free time, like Saturday and Sunday's extended working periods, or holidays. Scaling up it would make for 28-42 weekly hours, and an approximate of 120-180 monthly hours.

With this considerations, the desired cases of study that would be played were listed: from personal experience, reviewing a game would take an average of 5 hours, which would have a cautionary 2 hours added to integrate the analyzed content into the research. Together with the reference of completion time from the HowLongToBeat webpage, in which illustrative completion times are given for different types of playthroughs, it was approximated how much the chosen cases of study would take.

- *Fallout: New Vegas* (already played, but more research needed to be conducted; was analyzed during the planning phase)
- *Dark Souls* Series (already played and highly synthetized; was analyzed during the planning phase)
- *Façade* (7h)
- *The Church in The Darkness* (13h)
- *Road 96* (15h)
- *Night in the Woods* (16h)
- *Caves of Qud* (17h)
- *Outer Wilds* (22h)
- *Dwarf Fortress* (73h)

Parallel to playing cases of study would be the literary research, which would take the rest of available time. This would go from the minimum to reach the four daily hours to as much as personally motivated to go to, respecting breaks to optimize performance, while covering as much content as possible.

After the documentary phases were stablished, a priority order for the cases of study was developed by using the time they would take. This served as a dynamic contingency plan, in case the research's progress was halted by external causes: by starting the shortest games first, more content could be explored regardless of the real workload encountered during the literary research, while not leaving either parts of the project unattended.

Figure 1 Planning & Research phase Gantt chart (1)

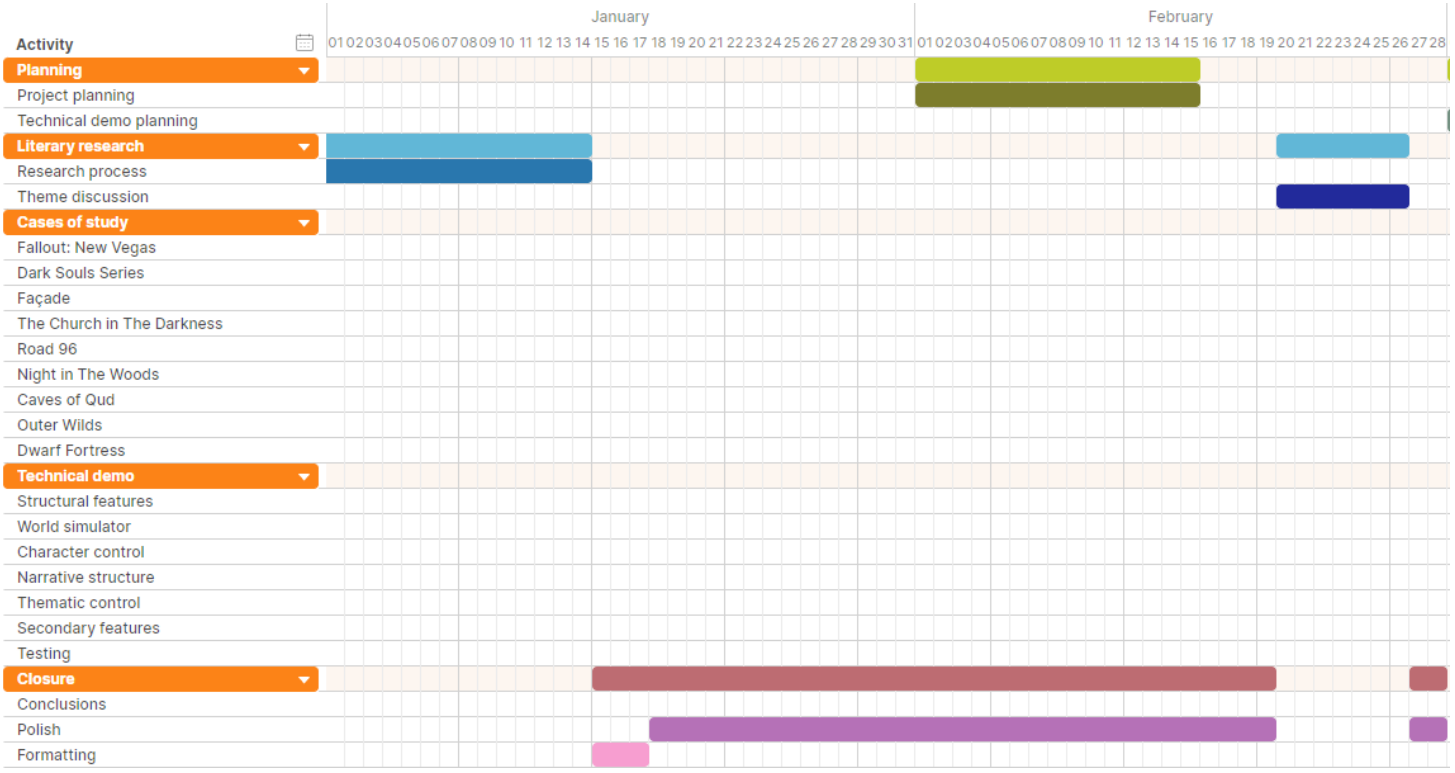
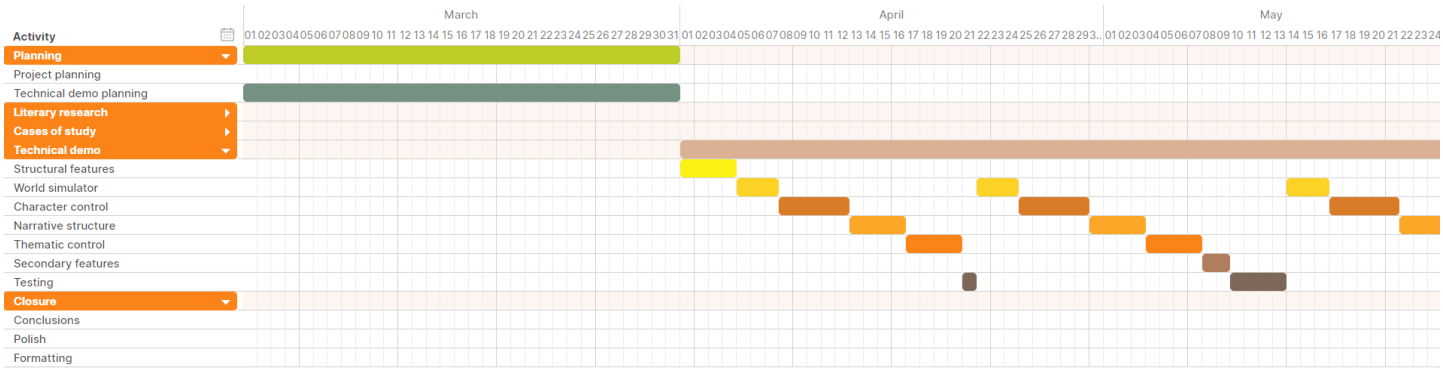


Figure 2 Planning & Research phase Gantt chart (2)

Once the information gathering was over, all compiled ideas would be merged into a report that would contextualize the project and provide the starting tools to develop the prototype.

For the technical demo, the following planning was done after enough information was gathered, so that the prediction being made could yield sensible results (this planning would be then expanded in *Trello*):

Figure 3 Demo development phase Gantt chart (1)



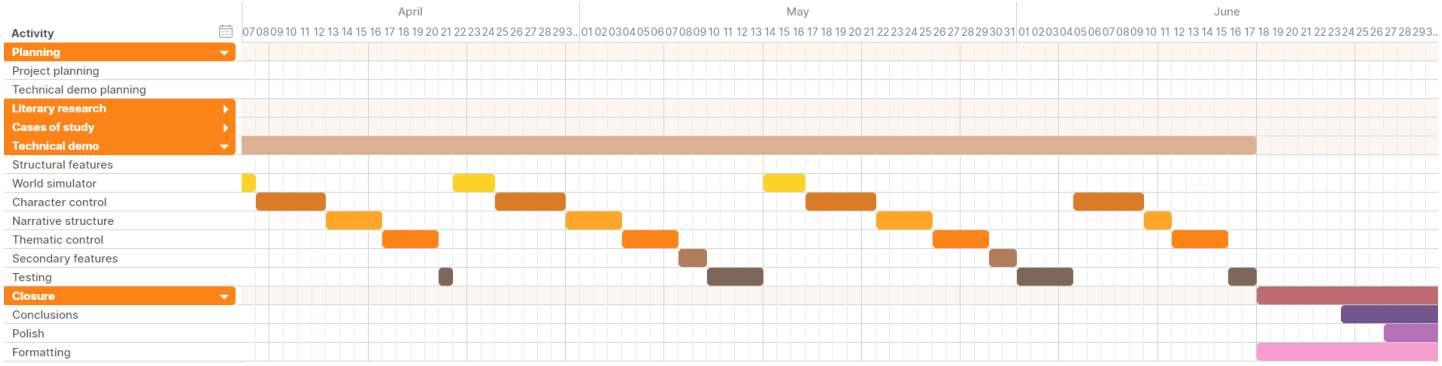


Figure 4 Demo development phase Gantt chart (2)

During this stage, an entire month was dedicated to figuring out how to merge all available knowledge into a solid system structure, developing a planning for the system's implementation. Each of those functionalities are properly explained in detail in the 6.

Technical discussion section, but in short, each iteration of the prototype was planned to go similarly to the following list[1]:

1. Event manager (full implementation)
2. World in one scene; world with objects
    a. World in multiple scenes; check of all world elements
        i. World scenes with connection weights
3. Theme base module
    a. Theme structure and player tracking
    b. Theme registration
4. Character manager, with communication manager-characters, as well as character's base structure
    a. Character planning, goals and behaviors; character-world replication; character relationships
        i. Factions; character joint actions
5. Narrative recording on a character basis; regulation of how many open narrative threads the program can handle
    a. Narrative structure formula computation, including strike system
        i. Narrative forced factors and setting control
            1. Narrative structure templates
6. Timer system (fully implemented)
7. Base debug structures
    a. Automatic debug structures
8. Tool state recording structures (save and load)
9. Game minimal set-up (first scene, with at least 3 characters and 5 events)
    a. Game expansion (quantitative increase)
        i. Game conclusion (added interactions and details)

2.3. Reach

While the aims of this research are focused on generating validated results in a very specific field (dynamic emergent narrative), it will also try to compel anyone with a basic understanding

[1] Consider each tabulation as an iteration of a feature

of videogame or interactive narrative knowledge, presenting an accessible but complete theoretical basis (both the theoretical framework and state of art), then going more in depth during the technical demo sections, and returning to a more general language for both the system explanation and conclusions.

This is not to say that the system is trivial to understand: precisely because the designed system is centered around a dynamic control of the world, a certain level of design vision and programming confidence is expected from those who might want to use it, which makes it so that complex patterns can be implemented without hindering user performance.

2.4. Scope

Considering how novel the field of dynamic interactive is both from a historical and personal perspective, it was not a guarantee that all systems could be implemented with the restricted schedule provided, and even less to the degree that it would be desirable. Aware of such situation, the prototype development was approached through an iterative process, to begin making each component of the system in its most basic form, then scaling it up to the desired results, as seen in the 2.2. Scheduling section. The main obstacles that could interfere in this process were:

- Slow implementation: the theoretic prototype is correctly conceptualized, but there is not enough time to implement it.
    - Knowing from the beginning the limitations it meant, the only reasonable measure to adopt was the iterative, branched system approach.
- Dysfunctional implementation: the theoretic prototype is correctly envisioned, but the specifically chosen structures are not adequate to fulfill their purpose.
    - The most realistic possibility is that, by the time this problematic is identified, it is too late to deal with it, due to the firstly mentioned limitation. If this point was reached, three alternatives could be taken:

1. Identify the problems and try to rework those aspects at the cost of a more developed prototype.

2. Acknowledge the problem, but continue the development to more accurately note why it failed and how it affected other parts of the system, as a possible starting point for future projects.

3. Ignore the particular sub-system, leaving the previously functional version in, and progress the development process accounting for the new context. Through this possibility we see another of the iterative process' advantages.

- Incompatibility: the theoretic prototype was not correctly developed, thus any attempt of implementing it yields unsatisfactory results. This would most likely be a consequence of the innovative part that it is theme-based simulation, paired up with trying to push most of the currently accepted conventions in dynamic narrative to the limits that the project's time constrain puts compared to the system's magnitude.

    - Despite the failure this outcome would represent in the scope of this project, it would be considered a positive outcome nonetheless, because from the beginning, there was no guarantee that the approach we would take was possible. Under this perspective, whether the system gets validated or discarded as a functional dynamic narrative tool is indifferent to the minimal value the research was developed to deliver.

2.5. Costs

Due to a null economical influx during the research process, costs were kept at minimum, resorting to getting material from libraries, public academic research and free tools. For the cases of study, we opted for free games, those already bought, those available in Steam's friend's accounts, and only paying for the cheaper few that remained inaccessible. Since all computing-related material was already acquired prior to the research, there were also no costs on that regard.

2.6. Tools

The following is a list of the tools used during the project's development:

- Researchgate: social network where most of the academic material was retrieved from.

- Steam: gaming service that provided easy access to a multitude of cases of study.

- Trello: website that provides an interactive framework based on iterative agile methodology, primarily used to control the pace of prototype's feature implementation.

- Tom's Planner: basic visual planning tool that was used to create the Gantt Charts with which the work blocks that compose the entire project were defined, allowing for high-level organization on extended periods of time.

- GitHub: hosting service where the research's multiple iterations are stored, both to see how the development evolved in comparison with the planning, and for the technical demo phase, where the repository would be stored in case that any revert is necessary. Overall, a very versatile and provident tool.

- Visual Studio: programming framework with which the most integral part of the prototype was developed and debugged (C++).

- PugiXML: library that provides structures to generate XML text, used for the creation of the multiple .txt files, be it for debug or to save the tool's states.

- PhysFS: library used to manage the multiple files generated during the tool's usage, ordering them into proper directories, and accessing them anew if needed.

- ImGUI: library that provides easily constructible interfaces that empower user understanding, and which rid the project of graphical burdens that did not contribute to the tool's improvement.

2.7. Results validation

While the final demo was centered on developing narrative possibilities for future authors, its objective was not to generate stories by itself, making it more of a technical quality check than a creative one. For that reason, its validation was more akin a checklist of features that it should be able to use functionally: as those attributes were described further in the research, they would be used as the criteria to judge the obtained results.

3. Theoretic framework

To properly contextualize the approach taken to the system's design, there are multiple concepts to be explained and discussed. The intention for this segment is to follow the preface, and from the start, establish the concepts that explain why the project was approached the way it was, and develop the reasoning to make sense of future decisions. However, this section of the research will only serve to discuss ideas at a theoretical level, based on the elements that are perceived relevant to the design process, and will not be directly reflective of any (practical) decision taken. That part will be saved until the state of the art has been fully presented (in Technical discussion).

3.1. The notion of art

The final goal of this research process is to provide an accessible tool to facilitate the creation of games through thematic elements. Essentially, this means the project is oriented to fulfilling somewhat subjective notions of art-making; thus to reason my choices, previous common ground has to be established, starting by the meaning "art".

The initial approach, prior to this research, taken on the understanding of "what is art" was to empty the concept from any notion, and build a definition by observing the common properties shared within multiple candidates. Such approach was mistaken, as it soon became obvious that art was not something to be found within the properties of ordinary objects, and instead in the perception of those, by human-like minds that could perceive art differently, thus invalidating the idea of the creation as the direct container of the artistic property.

With the consideration of the term as a subjective phenomenon, and after much struggle that led to physical properties being completely discarded, a list of perceived notions that could determine the concept[2] was made, and concluded that the one that uniquely defined art was "a

[2] One very similar in nature, but not quite in content, to the one that Gaut presented for his cluster theory of art (Gaut, 2000)

provoked state of meaningful emotionality"[3], albeit by a creation that is perceived to have expressive authorship[4].

While this categorization might not define a generic nor descriptive attribute of art engagement for other people, it is the framework within which notions of artistic capacity where judged and implemented. Since this definition adjusted to the development's perspective, and on a general sense it is not too far from the mainstream artistic view[5], only differing in scrutinized and thought-out details, it was deemed acceptable as a basis on which to start the research. Otherwise, the grounds that this project would have to cover would extend to the point of impeding any progress in any relevant aspects, due to the incredibly vague nature of the topic at hand. Moreover, even if we elucidated a more "correct" or "objective" take on art after the already conducted research, the notion of personal experience with art (the reason of this project) would not be altered, thus failing to bring any relevant information to the decision-making criteria that would be used to ultimately define the system.

Unfortunately, in order to translate this generalist mindset into a more practical videogame-centered context, certain guidelines of how it affects the creative approach would have had to be developed beforehand, which were not, and represents too much of a burden to develop alongside the more specific research this project requires. Consequently, such orientation will come from personal, but not developed, hindsight, as well as some notions that were derived in the aforementioned prior exploration of art. In any case, to get to discuss said criteria, we ought to firstly address various concepts and ideas that are more specific of a videogame-centric theoretic framework.

3.2. Videogame-centric theoretic framework

Before we tackle more technical concepts, there are two overarching constants that will be repeated through the entire of the research, and which should be clear from the start.

[3] The emergence of such feeling can also be described, depending on the connotations each perceives to be better adjusted, as resonation, rumination, trespassing the mundane, beyond oneself or transcendence. In any case, it does not relate to generic ideas of emotionality, nor it is descriptive of the intensity at which it is felt. In a sense, it can be tightly related to Friedrich Hegel's observations on art (Houlgate, 2009); and more loosely, to Aristotle's catharsis.
[4] That is, it's ought to be perceived as a creation intended to exist for the sake of existing, not just a natural act or as an aimless effort.
[5] The general view of artists, that is.

The first idea is narrative. Despite its normalized use, it englobes much more significance than one could think at first glance. Taking H. Porter Abbott's stance, Marie-Laure Ryan presents a definition of narrative such as any combination of story and discourse, the first being the events that take place in the narration[6]; the second, the mannerisms with which it is told (Ryan M.-L. , 2006). The simplicity of the definition helps englobing all the variety and novelty that surged in the field with the advent of both digital technologies and extra-literary practices, which sprouted multiple approaches to the use of narrative, categorized in Ryan's book based on their purpose:

- Practical approach: that which values the effect of stories, told through digital devices, in people's life, mostly disregarding their quality or meaningfulness, focused on studying their consequences.
- Metaphorical approach: usage of narrative concepts as mediators between a user and a non-narrative application, concerned with facilitating access and use to an unintuitive software.
- Expansionist approach: conceptualization of narrative as an ever-changing form based on the means authors can access to tell stories; since the rapid development of technology still goes on, it only follows that narrative has to transform to adapt to the possibilities of digital spaces.
- Traditionalist approach: importation of classical narrative conceptions into digital space, applying them faithfully despite new media possibilities.

To better understand the nature of the project, this first typology serves us to define it as a mix of expansionist and traditionalist approach, embracing both the novelty of the videogame media and how it needs to evolve from the typical conception of narrative, and the knowledge harvested from traditional narrative fields[7].

An even more relevant concept in videogames than narrative is the second one, interactivity, the property of letting a user influence the behavior of a system. From the combination between both concepts we obtain interactive (digital) narrative (IDN)[8], the process of engaging with a system to obtain a potential narrative (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017). For the

[6] Not the events that take place in the diegesis, but the chosen events being narrated within.
[7] While both approaches are theoretically opposed, the pool of knowledge and insight they both offer are not mutually exclusive, and will be used depending on the design needs, as the ultimate goal is creation (of art).
[8] The "digital" part will be omitted throughout this document because it would be redundant, since it will be the focus of the research at all times

sake of clarity, it would be relevant to mention how the authors define narrative interactivity in three possible distinct forms, which are not mutually exclusive between them:

- Text-based: most prominently defined by interactive fiction and hypertext, media that are defined by their pure text format (although it can include images in some cases).
- Cinematic / performative: media where the user interacts with audiovisual content. It includes videogames, interactive cinema, multi-linear TV shows…
- Ludic / experimental: interaction beyond classical conventions, most prominently featuring videogames' playability, but also the likes of interactive improvisation.

Common to all types are the components of the narrative, divided into the protostory, the static parts which conform the basis for the interactive scenarios to occur; the narrative design, which is the flexible, multiple presentations that can be adapted into the narrative progression, suspended and reformulated based on player action; and the narrative vectors, the elements that inform and limit player action to assist them in the realization of the narrative design.

The authors better exemplify the unique free nature of interactive narrative by abstracting it into a matrix of possibilities, where each game state produces an output based on a specific player input. Such matrix can be, when the interactive narrative is dynamically controlled, initially empty, as the results of a player input are unknown to the system too, and need to be generated when the input is introduced in the system's rules. When the narrative is not fully authored, the matrix can have hundreds of game states, as the player input's results will depend on minute case-by-case states, applicable to many actors, but different in output.

Interactivity serves as a lens from which to interpret narrative, but is only a possible way to understand it, and it coexists with others in a larger pool of lenses, or modes. A complete guide to understanding the multiple modes of narrative would be the following (Ryan M.-L. , 2006):

- External – internal mode: differentiation between materially encoded stories and those that are not directly signalized (for example, a narrative built onto the events of a *Sims* playthrough (ElectronicArts, 2000-2021)).

- Fictional – non-fictional mode: expression of imaginary events or real occurrences.

- Representational – simulative mode: story as a set of decided events or as a real-time simulated sequence. Representational mode could be a branching narrative; the key element is that the content is authored, not the means through which it is reached.

- Diegetic – mimetic mode: presentation of the story being told from outside the diegesis, through a narrator's intervention, or a direct showing of the events, a performance of the actions that conform the plot. A story usually has both instances of this mode intertwined between them, as it changes voice from the agents to the narrator, and vice versa.

- Autotelic – utilitarian mode: whether the story is presented for its own actualization, or it has a purpose beyond its own display.

- Autonomous – illustrative mode: distinction between cognitively[9] new stories and retellings or adaptations of previously, presupposed to be known, stories.

- Scripted – emergent mode: when there is a predetermined order of the story's events, in contrast with a dynamically performed or constructed plot line.

- Receptive – participatory mode: positioning of the audience as an observer or a participant. The participatory mode applies both at a discourse level or at a story level. Examples for each would be hypertext narrative and interactive drama, respectively.

- Determinate – indeterminate mode: consideration of the narration as a thread of continued events, or through one or more events dispersed enough that their historical continuity is only graspable through the significance of the known events, or is directly unclear.

- Retrospective – simultaneous – prospective mode: mode that defines whether the evens are presented as past, present or future sequences, respectively.

[9] We make the distinction between new and cognitively new to emphasize how the story must be seen differently, even if structurally, thematically, essentially… they are the same as others; it refers here to the sequence of events being the same.

- Literal – metaphorical mode: mode that defines whether a narrative fulfills certain characteristics of the narrative definition, and if not, how similar it is to a composition that has a complete property of "narrativity". This mode refers to a proposal of narrative definition made in the book, but which this research is unconcerned with[10].

Additionally, distinctions for interactivity are also provided (Ryan M.-L. , 2006):

- Internal – external: the acknowledgment of the player as a member of the story, or as an external force that influences it.
- Exploratory – ontological: interactivity limited to broadening the player context in regards to the world, or that which allows for consequential narrative actions.

To finally conclude this section, and before advancing onto more specific topics, we can superficially define the system generated during this research as an internal or external, ontological interactive narrative artifact[11] of external, fictional, simulative, mimetic and participatory mode, with the authorial-defining modes being open for choice.

The one category that could cause confusion to be defined as authorial-based is that of scripted – emergent; while the system is thought to sustain emergent narrative, it could be used under the control of a larger, predefined narrative order, as one of the design objectives was to allow creative freedom and flexibility.

3.2.1. The nature of agency

That the media of videogames is an interactive one escapes no one's perception, but there seems to be an erroneous, widely spread notion regarding the role that agency plays within it. Agency is described as the capability of the player to take informed decisions, the consequences of which can produce a range of differently meaningful results (Mateas M. , 2001) & (Murray, 1997). Those results, in order to be considered meaningful, require a certain level of interpretation from the player: Mateas himself further clarified that agency partially occurs, too, in the player's mind (Hammond, Pain, & Smith, 2007).

To put an understandable example of what this subjectivity entails, a speedrunner might complete a game 2 seconds faster than previously due to their informed decisions, which would

[10] For the sake of it, our project fits into a literal narrative.
[11] Artifact taken as the elements used to create and experience the artwork, in this case, a part of it (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017).

imbue a powerful sense of agency to such result, while someone completing a game twice, with a different ending in each playthrough, even if derived from informed decisions, might feel that said difference is meaningless, because it does not provide an alternative they would realistically choose, thus missing a feeling of agency because the game was designed outside of their desired action paths.

Agency is so malleable of a concept that it can even be perceived in cases where it does not exist at all: if a game had a completely streamlined route of action, where there was no decision to be made, and the player was part of the system only to enforce the actions, there would undoubtedly be no agency present. However, if this set of actions were reflective of the player's intentions, agency could be perceived, because at no point the player's sense of choice is being impeded, and at no point the player would have to question whether there were more options available or not, because their preference was already available for execution.

We can even get very deterministic, and assert that no agency truly exists, because for an informed decision to be taken, the player should be able to freely choose the action they would like to pick, thus they are not deciding on their own terms, just following the rules and dynamics of the already limited game code[12].

At the same time, on a first-time playthrough, it is very difficult to not feel a certain degree of agency, because the lack of gameplay internalization and narrative knowledge gives the impression that the player is experiencing a result that can only happen through their decisions (even if in truth those are unchangeable, predetermined actions).

Beyond first contacts, agency can be actively regulated by design elements such as level layout, narrative presentation, lighting, balancing… (Harrell & Zhu, 2009), conditioning player action, and doing so almost to an intrinsic level, whether the game designer had planned for it or not, because at the end of the day, there is the aforementioned weight of agency in the player's mind.

In summary, while the definition of agency is a solid one, and has value when designing a game, agency from the player's perspective is multi-layered, defined case by case, and not necessarily correspondent with the system's designed agency (William, Harrison, Ware, Cardona-Rivera, & Roberts, 2012).

[12] The phenomenon is called illusion of agency, but it was not delved in, because there is not much value to be found, as it is a consideration that denies agency just because there are limitations; in a reality that is fundamentally limited, all agency is limited.

3.2.2. Agency from the ludology and narratology point of view

On top of the multifaceted nature of the subject, there also seems to be a typological problem, which is where the classic ludology vs narratology conflict, one that has been part of the videogame sector for almost thirty years (Koenitz, 2018), becomes relevant.

To put it in simple terms, ludology is the approach that defines videogames as a media of play, shaped by the activities the player engages in; meanwhile, narratology defines videogames as stories, usually examined through the conventions of other media's narratives. Neither of the two truly excludes each other, they refer more to how games are explained. The reason why they cannot be proposed as defining theories of videogames, in our opinion, is quite simple: videogames are diverse enough that they can have purely ludic games (which can be analyzed as narrative constructs, like *Tetris* (Murray, 1997)) and purely narrative games (like *To The Moon*, where the game's backbone clearly is found in the story's telling, with the player actions only being necessary to progress the system's instructions, but having such interactions nonetheless).

The fact that both styles of games can be commercially successful already indicates that, even if they are not experienced through an artistic standard[13], both paths have been validated to have a place within the media, and a potentially successful one, at that.

Having said that, we can address the mentioned apparent problem of typology, which is the existence of ludic agency and narrative agency. It was not enough to be subjected to player perception, that whatever agency we can implement can be directed towards the mechanics[14] or the narrative, with different players processing each more or less positively, fragmenting even more the effective incorporation of agency into the game's systems.

3.2.3. Agency's value

To top this whole discussion off, and returning to the first statement about the videogame-centric theoretic framework, videogames are interactive, yes, but they do not need agency to

[13] That is, if they are enjoyed without really achieving a state of meaningful emotionality, but nonetheless being positive, well-regarded experiences.
[14] The building block of videogames, the actions (player-generated or not) that define changes of state in the game world (Hunicke, LeBlanc, & Zubek)

work as potential artworks. As an example, we could take the entirety of the *Ace Attorney* saga, where the player cannot alter the course of the narrative, and any attempt of not performing the objectively correct actions will either be a waste of time with no consequences, or will be penalized (Capcom, 2001-2015).

All of this is to argue that we do not depend on implemented agency to craft a meaningful interactive experience, even if most games include at least one type of agency[15]. In the same way that we do not have such dependency, including it is not a guarantee of improvement, at least not for every case; a high level of freedom and choice implies more design complexity, and it can even bring some undesired thematic implications, making it not always preferable (Harrell & Zhu, 2009). For example, it is a common criticism to early open worlds that they feel empty and boring, despite providing a huge agency to explore and live meaningful interactions in whatever order the player decides.

The reason why so many emphasis is being put in agency is because it is one of the many tools designers have available to create, and while very powerful, it is no saving grace, and thus, should be understood precisely before using it as extensively as this research's system does.

Once that is settled, we can go on to see what are the potential benefits of agency, and what we attempt to achieve when making a dynamic narrative manager, which is a functionality that clearly aims to a more choice-driven experience. For ludic agency, it is relevant to:

- Consciousness: having to take relevant decisions means players are more prone to think about the systems and the effects of their actions, which can potentially lead to more informed, personal gameplay.

- Adaptability: understanding that there are multiple valid approaches can lead to a change of playstyle that redefines how to tackle the game's challenges.

- Planning: if the decision is perceived to have long term consequences, it can force the players to set goals and prepare certain aspects of the game in advance, expanding the playability to obtain better results.

[15] Albeit it is discussable to which extend videogames need perceived agency (those were the player feels like they "fit" in their role, when in reality there was only one role available to begin with) to have meaningful experiences.

- Mastery: since a meaningful decision is perceived as a non-obvious choice, acting with agency, whether the results are positive or not, enhances a sense of mastery, the process of internalizing the mechanics.

In terms of narrative agency, it affects:

- Immersion: as long as it reflects player intention, seeing the character do what the player would have done reinforces the magic circle. It affects both the process of decision-making (being able to do what is desired to) and the action's consequences (things going as planned).

- Engagement: being compelled to act means the player has a chance to put their capabilities to practice, driven by a want of competence that prompts them to act and take the initiative (Legault, 2017). Incidentally, seeing the consequences of one's actions is a way to obtain feedback to better define the capabilities of the acting agent.

- Expectation: meaningful actions, for the most part, spark consequences that mark the separation between two narrative beats, thus giving the expectations of change and progress.

- Tension: again, meaningful consequences mean an advance of the status quo, which following the traditional narrative models, implies a rise of tension (or the implicit advance towards a tension peak).

- Emotionality: although not necessary, if the player has an emotional investment, taking meaningful action means affecting or preventing an effect on the relevant entities, which in turn potentiates all other benefits of agency and generates more attachment.

- Introspection: while traditional structures allow introspection of human nature through the observation of others, meaningful interactivity directs the responsibility of the decision and sense of the experience towards the player.

It is in this last property that lies our interest in agency. Not all videogame's treasured experiences[16] have to be derived from how much fun[17] an in-game decision is, and instead, they can also come from moments where risky decisions had to be made based on personal moral standards, like having to endanger the diegetic well-being of a character to preserve the player's sense of integrity. Where other media can showcase those kind of situations, and they can make one reflect on the presented, intricate nuisances, videogames have the unique property of letting the player fail, or succeed, or face whatever gray area in between, without losing all of the authorial depth that characterizes the beauty of other media.

It is for this reason that there was a particular interest put in developing a system that supports strong agency: a game can be constrained to voicing authorial intent, and it can work nonetheless, but it would be desirable, in our eyes, that the industry grew in an environment where players can realize themselves in the more personal (thus, meaningful) ways possible.

After establishing the design mindset, there is still a crucial piece necessary to understand the approach taken: how to mix agency's potential with the system. Since the system is focused on controlling narrative, and is supposed to serve as a base for game development, it only makes sense to prioritize narrative agency. This is not to deny the introspective potential of ludic agency. Examples of such would include the fascinatingly complex web of minute intrapersonal details that high level fighting games players exhibit, the multiple expressive approaches that an oppressive environment like *Dark Souls*' can cause, or the unending decisions that *Minecraft*'s freedom allows.

While this approach was taken very early on, it would by no means solve many of the questions posed by the system's functioning. As comfortable as it is to stick to what works, which is traditional narrative and its mechanisms to craft emotionally meaningful situations, the dynamic part ironically shows the cracks of this same foundation precisely because it is trying to be used in a digital, interactive context, with the most prominent problem being authorship vs agency intentionality.

Such conflict is an obvious consequence of the interactive nature of games: narratives are usually not realistic sequences of events, just plausible ones, arranged in a specific way to fulfill a preconceived purpose. When there is an agent within them that knows not such purpose, nor

[16] Which directly ties to meaningful emotionality
[17] Ludic fun, as the concept of "fun" can be extended to any kind of enjoyment, for example, based on LeBlanc's typology.

how the sequence is supposed to follow, it is only natural that discrepancies would surface, even more when said agent is supposed to be the main actor. Moreover, if the notion of agency is that of informed meaningful and consequential decisions, it is contradictory to limit the narrative interaction, and efforts should be made[18] to provide a space where the player is not obliged to follow the designer's guidelines; it should not be about imposing a story, but creating a space that can sustain a conversation between players and designers as equally mature minds, something akin Ben Samuel's shared authorship (Kreminski, 2020). At least, that is the ideal strived for the media to reach in this research.

To summarize, we will focus our system on narrative agency, since ludic agency is too tied to the design core to be alterable by a secondary, relatively generic system. As a natural consequence, we will adapt the system to the needs of digital interaction to avoid pre-built ludonarrative inconsistencies as early as possible.

Luckily, there is a type of narrative uniquely capitalized by videogames that solves the discussed constrains, and which has been used for decades. To illustrate it, let us take the *Pokémon* franchise, an RPG that, for the case, can be simplified as the player using a team of at most six creatures to defeat opponents using a more complex version of rock-paper-scissors (GameFreak, 1998-2021). Because the games were never especially difficult, and a big part of their charm laid on the creatures' likability, mostly every player creates their team based more on their preferences rather than competitive viability. This kind of set-up easily creates small stories unique to each player, and any gameplay choices that are taken (for example, changing a Pokémon for a stronger one, or finding out how to evolve a creature that had been carried unevolved for many hours) become significant parts of that story because of the emotional relationships the player develops with said creatures. The essence of making a choice that influences the actors of the story, with such decision being a consequence of player evaluation of their game performance, is the perfect union that defines the next term we will take a look at: emergent narrative.

[18] Again, in order to provide an agency-based experience, but not as a general guideline for all games

3.2.4. Emergent narrative

For a more formal postulation, emergent narrative is defined as an application area of computational narrative in which stories are generated bottom-up by autonomous characters' behaviors (Ryan, Mateas, & Wardrip-Fruin, 2015).

It is worth noting that, similarly to agency, it is no definitive narrative solution, as it usually suffers from a lack of control and pacing derived from the lack of author restrictions. Nonetheless, an advantage it presents over other narrative approaches is that, since it usually uses the mechanics of the game to exert its emergent component, it does not present that much of a significant separation between gameplay and narrative, furthering immersion and diminishing the feeling of interactive limitation, as narrative interaction is expanded to the whole set of abilities shown to be available within the diegesis[19].

On paper, emergent narrative was immediately presented like the best take to bring an introspective-friendly context to the player. A potential problem was noted, however, when examining how well the primal artistic objective, that is, of generating meaningfully emotional experiences, was kept in this kind of systems. If we go back to the *Pokémon* example, we can see that choosing the trainer's team is an agent decision (not defined by game autonomy), so it has potential to generate introspection, pointing towards the possibility of meaningful emotionality.

However, for games with emergent potential like *Fire Emblem* (IntelligentSystems, 1990-2019), where the player has to choose which members of the army to send to battle in each map, such layer of introspection does not exist, at least not in this decision. This is because there usually is not the same kind of emotional attachment: diegetically, in the first case we either bring our pets with us on our magical adventure, or we relinquish them in the PC's box system; in *Fire Emblem*, not bringing a character to a battle means they just stay on the bench, waiting in case any of your units dies. Additionally, *Fire Emblem*'s permadeath mechanic limits player expression (even if it raises the significance of ludic agency manifold) due to the consequences of their battalion line-up being irreversible and punishing. In essence, we have two systems that function very similarly, but which have vastly different connotations both from a high-level design and player experience perspective.

[19] This is in contraposition to games where the only narrative interaction is through dialogue boxes or similar, and where player actions are dismissed and ignored outside of their area of use, very usually breaking immersion.

This is a good example of emergent narratives not implying meaningful emotionality, but because it concerns the framing of the game and how the play is contextualized, it is not something that can be controlled by our developed-in-a-vacuum narrative manager. As a consequence, it will be left to game developers, and we will not take it into consideration.

Going back to the *Pokémon* case, another problem seemed to surface: the question of how much of it really is a story, how much is just a consequence of a specific system of play, and how aware (or how much it is being felt) is the player that there is a story unfolding. Even though humans are pattern-searching creatures, and as such, it is not difficult for us to find sequences in unintended places[20], it is questionable to which degree we should expect the system to define and contrast an on-going narrative against the moment-to-moment events of the game world.

This, among many others, is one of the problems that were exposed to be intrinsic of emergent narratives (Ryan, Mateas, & Wardrip-Fruin, 2015):

- Modular content: referring to the elements that are used to display the game states, the challenge resides in being able to convey the current state of the story through enough means so that it is understandable despite the complexity and diversity of contexts the narrative can afford for. An example of a game that navigates this issue finely is *The Sims*' state bubbles (ElectronicArts, 2000-2021).

- Compositional representational strategies: as a result from the system having to consider large amounts of contextual data to give results, it is easy to generate bad code, both dysfunctional and unreadable, to try to control very differentiated aspects of the game's reality. The difficulty, then, is not only solving this problem, but designing structures that are functional in multiple contexts, and applying generic rules that predefine the result of each interaction, decreasing author burden and expanding the system's capability to respond to narrative input.

- Story recognition: the process of identifying that a storyline has begun, in order to ready it for processing, controlling and development. The difficulty in this case is being able to identify what is going on in the game world beyond the parametrized data

[20] As we previously mentioned, even a purely mechanic game like *Tetris* got its own narrative analysis (Murray, 1997)

(spokenWithX = true and questProgression = 3, for example, codifies a certain scenario), and understanding what kind of story is emerging, to further develop it later.

- Story support: feedback that informs the player of changes experienced by the world and its characters, not with the purpose of forcing a narrative, but instead, showcasing possible interest focuses. The nature of such changes can exist in any moment of time: those that might take place, those that are occurring, and usually less interestingly, those that have been completed. As an example, the authors present *The Sims 2*, where characters have visible aspirations and fears, which informs the player of what narratives they can work to achieve.

Before taking care of those challenges, however, we must first return to the initial question of the granularity we should expect the system to generate and define stories with, which by necessity calls for a definition on "story", and what it is not: without knowing so, there is little we can work with in the realm of storytelling. It is worth pointing out that because we are working with a field in constant development, and with many unknowns, some of the traditional conventions about stories will have to be taken with a grain of salt to adapt to the options offered within videogames, as well as filtering the conclusions through the aforementioned approach of artistic results.

3.2.4.1. The nature of stories

The definition of a story most relevant to this research is one describing it as a succession of state-event-state changes (Stein, 1982) & (Gervás, 2009), that is to say, that the central element of story-telling is that of the alteration of the status quo, generally focused on how, who and / or why it is changed, what is changed, and which consequences does it have.

It is under this conception that conflict gains relevance: while it is not necessary in order to have a story, change usually comes from or derives in conflict, be it internal or external. On top of how well this blends with the form of storytelling, it also implies the contraposition of two confronted elements, which easily derives into situations of duality, that allow contextualization and development of said conflict, thus not only serving as the base, but as the body of the story too.

Conflict and progression alone, however, are not components that directly lead into an artistic experience. The immediate next layer is to achieve an engaging story, that is, construct it in a way that satisfies the cognitive demands, as varied as they can be, of the audience[21]. While this step should not be overseen, and in fact will make a big difference between what puts the, in our case, player in an assertive state to experience the game, an engaging story also does not reach meaningful emotionality by its engagement alone. It is emotional, by definition, but it does not reach the intended meaningfulness.

The truth is that there is no real way to create universally beloved stories, as we have cognitive preferences, and multiple reasons to interact with stories. Despite that, there is an element that, if correctly appealed to, serves to satisfy any drive to listen: our own humanity.

The human condition has certain innate tendencies that stories universally cater towards:

- Pattern recognition: evolution resulted in brains built to figure out repeated elements, no matter how abstract they might be. Since stories tend to require an extended period of time to experience, the likelihood that we begin finding more than what is beyond the surface increases, generating a game of text and subtext that increases engagement.

- Novelty: most of the times, the stories we go through are not made by ourselves, and have elements that we were not aware of or did not think about, naturally drawing us into the fantasy.

- Communication: no element of reality[22] can be fully conveyed through communication, as language (our primary, and many times most accurate, form of communication) refers to our grasp of an object, not its real nature. As a consequence, we naturally abstract any received information and internally compute it as a specific idea or sensation, based on multiple factors that go beyond the spoken element (semantic and social context, tone, timing, cultural baggage…). Being able to contrast, share and acquire new ideas with other entities is an inherently desired process easily achieved through narrative.

[21] It is important to note I'm not saying that all stories should be explosions and mindless violence. Cognitive demands expand to information retention, information presentation, esthetic style, audio emission, dialogue style, scene or action pace… The cognitive demand is satisfied by anything that we physically add to present the story found in the creator's head as a consumable experience
[22] With the possible exception of words themselves

- Empathy: to better judge the social contexts we are in, humans have developed an instantaneous response to other's showcase of emotions, altering one's own, diversifying the emotional landscape we can explore (at the end of the day, another kind of communication) without the risk of being hurt.

What is more interesting is how those human characteristics blend together: let us take the *Pokémon* example once more. The protagonist trainer encounters their rival, and they begin a fight. This and any set-up, by itself, has no value, and it is only when a human-like mind perceives it that we can expect meaningfulness to appear. Given that it is not the first battle, the fact that the player is fighting is probably not very important, because it has been happening through all the game, so it is accepted as a "natural" event more than a meaningful one[23]. What is more, as engaging as physical conflict can be, it is not a very emotional subject by itself: only through its context it can make us feel fear, anger, happiness, sadness, surprise, excitement, arousal, rejection… It is the context that matters. And what is the context? On one side we have our Pokémon friends, with which we are supposed to develop a fantasy of companionship, of union and camaraderie. Now those are emotional concepts, the themes, that which appeals to the human condition, the abstractions behind our actions, or as Robert McKee puts it, the archetypal experiences, as the opposite of the stereotypical (McKee, 1999). On the other side, we have our opponent, our rival, a figure that for those who entered at the beginning of the saga, was presented as an element of competition, of frustration, challenge, a call for self-affirmation. It is mostly through the perception of those two elements, which is built by how they are presented in the story, and by what opinions we might have had about them beforehand, that the mood and player emotional response is built on[24].

[23] It is important to account that such would not be the case for someone who, for example in this case, strongly empathized against animal abuse due to their emotional, personal experiences with the topic. Any element we introduce in our works has potential to attract and reject members of the playerbase, which is another reason why only the parts that strengthen the final vision should be included.

[24] We say mostly because we acknowledge that engagement has a part to play in this. However, we can lack engagement and still enter a state of meaningful emotionality, but no matter how engaged we are, we will never feel that way if there is not an abstract, humane core that builds on it (technically, the case could be given where the player forms those values around the game due to personal circumstances, but from a design perspective, it would be inefficient to assume that as a central pillar of the experience).

It can be said, then, that the core of story-telling lies in creating a world capable of sustaining a sequence of circumstances that are to be in-taken as meaningfully emotional experiences[25]. Citing Robert McKee again, story is metaphor for life (McKee, 1999, pág. 27).

3.2.4.2. Stories and interpretations

As helpful as this understanding of the story-telling art is, it does not seem to clarify how we must envision the ludonarrative conflict in our system, that is, the aforementioned blur between ludic play and narrative play, as to distinguish what parts of player interaction should be computed as emergent narrative.

Arguably, it is undeniable that *Tetris* has no plot. However, Janet Murray is right when pointing out its narrative themes (Murray, 1997). If we examine *Pokémon*, its emergent narrative is clearly distinguished due to its integration to the plot, that is, the journey of the character to become The Champion needs the "trainer growth" subplot to function and to be concluded. *Tetris*'s case seems different because it lacks plot, but essentially, the keys are the same: player actions alter a state into another. Even if we take the game at face value, there clearly is a story about blocks trapped in a continuous descend, being cleared out to make space for those that will come after them. And this cycle easily parallels many aspects of our existence. Precisely because *Tetris* is such an abstract playground, we could make hundreds of narrative interpretations, and while we could not affirm with certainty whether there was a narrative purpose or not, it would be undeniable that the story was there.

Respectively, we cannot assure that *Pokémon*'s emergence was intended: we know it was based on the RPG genre, and that the collecting and trading elements were central (DidYouKnowGaming?, 2019), but we know not whether the emotional element of team composition was a planned story, especially with how nebulous the concept of emergent narrative was at that point.

The truth is that, especially in any dynamic media, every sequence of events can be read as a story, whether the one experiencing it perceives it as such or not. The only thing the author can control is where the focus is put; in doing a system to control dynamic narrative, we can assume that there will be a narrative focus. In fact, this fluidity present in story definition is a positive

[25] Again, with the meaning of emotional being closely related to humanistic, archetypal, transcendent, out of the mundane…

to the system, since any structure that relates events is sufficiently good to hold a story, and it allows the possibilities of the narrative space to remain open to any future developer's storytelling style, whether they like obtusely presented plots, or very detailed ones.

While this property empowers the designer's freedom, there is still a significant disconnect between what will be described as a story and the mechanic's system, in the form of classic narrative structures contraposed to ludic emergency. To reconcile both sides, we can use an element present in both aspects of storytelling.

3.3. Theming

To summarize the train of thought until now, the system aims to create meaningfully emotional experiences, and the best way to achieve it in videogames is assumed to be through agency's narrative introspection[26] and story's emotional cores.

Those cores are, as it has been mentioned multiple times already, the story's themes. Being the foundation of emotional conflict, they also fulfill a very practical need, of having a common denominator among story beats, independently from narrative contexts. This capability of summarizing a sub-story's essence in a single object was seen to come in handy when having to deal with sending information about the world state and how it should progress, which only reinforced the solid base that themes represented as an essential element of story dynamic generation.

Even though the commonality of narrative meaning that themes entail was envisioned as a central element to all sub-systems developed in this research, themes can appear in many ways, and to be conscious of the effort that would have to be ideally put to accomplish a full implementation, a typology of themes is necessary.

Prior to that, it must be understood that, in essence, themes can be anything: by being an abstraction of concepts morphed into an element of a narrative: as long as such element is relatively recurrent, it can be identified as such. And because narratives can be represented as looping structures of state changes, and especially in videogames, where concepts like game loops are central to many design approaches, it is not unusual to find elements constantly

[26] Desirably in tandem with gameplay introspection, but that is out of this research's scope.

repeating, intentionally or not, through a narrative's length[27]. Trying to encapsulate the most prominent types, while adjusting to the broadness of the topic, two macro categories were devised from prior analysis of multiple cases of study.

3.3.1. Theme typology

3.3.1.1. Framing-based themes

Though the most common interpretation of themes can be found in how characters act, themes expand beyond that, without necessarily leaving the diegesis, although not limited by it. Framing-based themes englobe the different types that generate from having a prior and / or posterior context that relate within, working on the most general levels of the narrative:

- Artwork structure: themes informed by the division, order, presentation… of the story.
    - A classic example of thematic artwork structure is Dante's *Divine Comedy*'s reflection of its essences through the formatting, at every level he could, of the poem's building blocks (Mambrol, 2021).
- Contrast: comparison between two or more different perspectives on the same theme, be them opposite or relatively concurrent, to form a more informed final result. This is one of the most typical, as it is enough to have an occurrence A essentially[28] relate to an occurrence B to achieve it.
    - Taking *Celeste* as an example (Thorson & Berry, 2018), the contrast between Madeline and her other self, which for this case can be considered as different characters, can be taken as an exemplification of the conflict between the safety of mundanity and the confidence that humans can develop in it, and the awareness of unknown, possibly hurtful things, lying in wait outside that fragile personal space.

[27] This innate repetition being one of the many sources of theme noise, and especially in videogames: if the core game loop is thematically detached, the narrative can easily feel more distant, but if it contradicts a theme, it can express the opposite of what it initially meant to, and push against the storyteller's intentions.
[28] Essentially in the sense of "in its essence, not superficially", as the abstract core of a theme.

- Re-contextualization: contrasting theme, done within one same element in two or more different points in time. While functioning the same as a contrast theme (for example, if a character does A, learns something, and is compelled to do B) it is much stronger because the subject focused on is the same, thus the contrasting difference is usually much clearer.

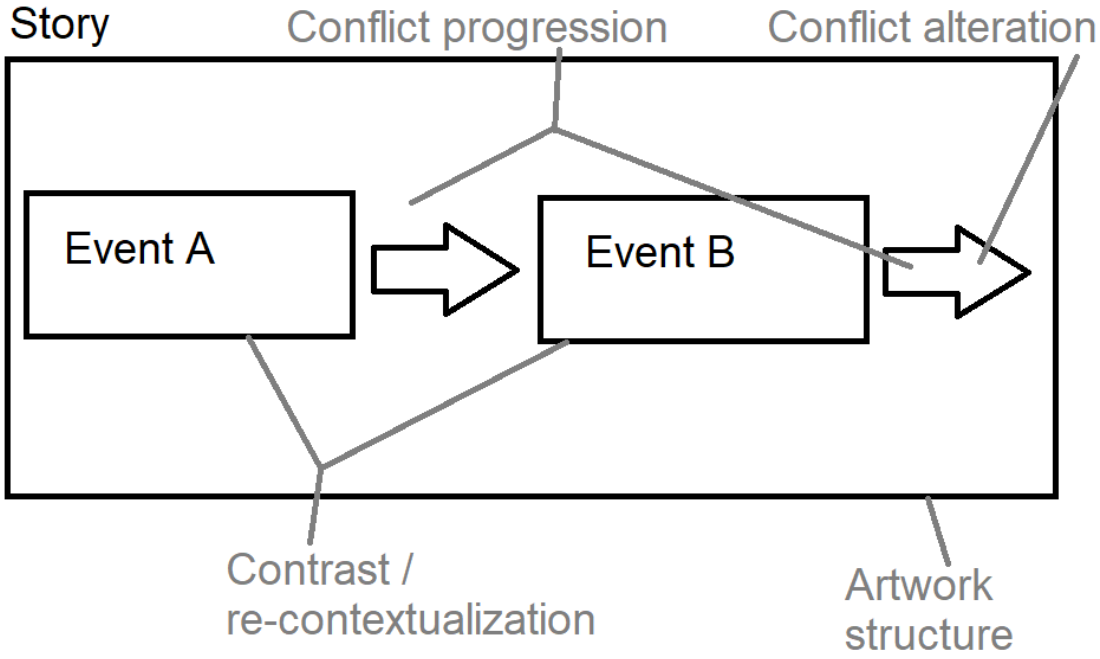
- Conflict alteration: theming found within the process of progression of a conflict development, as defined previously, a change between (meaningful) states. One of the most defining sub-types would be conflict conclusion, as it not only has meaning by itself, but it weights much more because it is the conclusion to the thematic artifact. It differs from contrast in that the meaning is not found in the differences between states, but in how the transition is done.

    - Continuing the previous example, spoiler free, the conclusion of *Celeste*'s character conflict informs the previously established theme, using mechanics and emotional relief to enhance the conflict's thematic conclusion. Note how this facet can only be grasped from understanding the process of state change; contrasting the Madelines before and after the conclusion, or even the tensed and relieved conflict, would not yield the same insight than the observation of how it is solved.

- Conflict progression: themes informed by how different facets of a story element (be it plot or other themes) are sequentially presented and / or related; in other words, those are not about how a theme is carried through each instance of the narrative, but about how the presentation of the progression informs the theme. It is not unusual that a theme self-references a particular aspect of the essence it represents by how it is presented to the audience.

    - In *Shadow of the Colossus*, the repeated act of killing one colossus after another, while being a well-stablished videogame convention in the form of a "boss rush", is intertwined with the narrative in such a way that it can be interpreted as the main emotional indicator of the moral connotations the act carries.

Figure 5 Framing-based themes



3.3.1.2. Element-based themes

Categorization that englobes the moment-to-moment thematic expression.

- Actions: a particular action, whether it has impact on the plot's state, conveying a theme.

- Character setting: a character's personality traits, physique, state and relationships, their individual attributes in a vacuum, all can contribute to striping them down to their thematic essence.

- Environment setting: similarly, an environment's weather, lighting, structures, the objects that can be found in it, the relationships between the fauna and flora… can also denote thematic elements[29].

- Narrative elements: the micro-level presentation of extradiegetic elements is also a telling factor; mainly through the use of rhetorical devices like hyperboles, parallelisms,

[29] Beyond, of course, the tonal function those elements can provide; again, a property we will not focus on in this research.

cosmic irony, references, metaphors, repetition (leitmotivs), symbolism…, but also through elements like the narrator's voice.

- An example of evolving symbolism can be found in *The Garden of Earthly Delights* (Bosch, 1515), where the element of birds is firstly introduced as a friendly source of happiness to humanity, but later on, being the punishers of hell. Due to its context, it has been suggested that birds are, then, a symbol that represents sin through visual metaphor (GreatArtExplained, 2021).

Figure 6 Garden of Earthly Delights, divided in Paradise, Earth and Hell

- Esthetic: another classic source of theming is the visuals, which can go from character design, to the chosen pallet and lightning, to how the pallet is treated.

  - An example of smart usage of a thematic pallet can be found in the anime *JoJo Kimyou na Bouken* (Araki, 1987-current), where the nature of justice is recurrent. In the scene in question, the antagonist has the upper hand, and argues that justice is defined by the victor, in classic historical negationism fashion. The pallet is then swap, literally reversing the depiction of the scene, to foreshadow that the justice that will prevail, thus the victor, is that of the protagonist.

Figure 7 JoJo Kimyou na Bouken pallet swap

The essential difference between framing-based and element-based themes is that the prior is informed by a collection of the former, and its meaning resides in that set of relationships, while element-based themes are self-contained, and handle a more specific reading of the overall concept on their own, even if it ends up being the same, tangential or opposite to the final conclusions a reader can extract from having a bigger picture.

3.3.2. Player theming

With this being a videogame-centric research, a big focus was put on player action and its theming potential. As complex as the relationship between player and game interaction can be, however, from a narrative standpoint, interactivity can be directly interpreted as an element-based action theme.

Additionally, the reach of videogame's thematic expansion extends to agency too, and how the alteration of agency can be used to imply meaning (Harrell & Zhu, 2009). This is no different from a framing-based conflict alteration theme.

This understanding of player action and agency as already fitting narrative possibilities not only expands the domain of thematic richness, but it also reveals, at least conceptually, a very effective solution to ludonarrative disparity: by using themes as a cohesive language between both aspects of a game, and using them as the building block of the story, the conflict that emerges when they are developed separately disappears.

Taken at face value, the connection might seem quite conclusive, but of course, reducing the media pillars that are player action and agency to a simple subcategory within a typology would be reductionist. To gain a better grasp of what this proposition entails, different aspects of player interaction should be explored, highlighting those that gain expressive power due to the specificity of the media:

- Mechanical motivation: the incentives found in the world for player action; not so much what the player has to do to progress, but what the designer intends to condition them to do through the creation of the mechanical conflict.

    - *Dark Souls*' stain mechanic, that pushes players to go face the challenge that killed them, is a thematic found in mechanical motivation, as it enforces relevant concepts of the artwork, such as struggling, perseverance, overcoming, but also cruelty, failure, torment, instinctual thoughtlessness or regrets, in case of failure.

- Play performance: relevant to both mechanics and dynamics, it refers to player's intrinsic development, be it in execution or in cognition, and what is implied by said improvement. It is necessary to keep in mind the significant difference between an interaction that is given to the player in isolated narrative contexts, and one that is achieved, even if it will be seen only a few times, as a result of player adeptness, be it tied to the game's context or not[30].

    - In *Pokémon Snap* (HALLaboratory, 1999), for example, being able to decipher relatively complex chains of reactions to achieve a situation worth photographing is one of the primary ways of developing the themes about connection and coexistence with nature.

- Mechanical progression: be it acquiring new mechanics, losing them or developing previously existing ones, the progression of play possibilities becomes central to describing character's effect on the world.

[30] The first is a system concession, usually to allow specific moments to occur (use the final McGuffin to end the story is probably the most classic one; it can have thematic depth, but it is not using play performance to present it); the second is a high-level understanding, a culmination, of a (or what should be a) well designed, thematically coherent system.

- Mechanical limitation: changing the rules of play under specific conditions brings attention to the essences that cause such divergence, something much more noticeable in game due to their ample focus on player agency. It is distinguished from mechanical progression because it is defined by the spatial position of the player, and thus, is assumed to not be a permanent change.

- Mechanical relationships: relating mechanics is, intrinsically, a form of framing-based themes, the more extensive (and hard to keep coherent) the interactions, the more information being related. It is a concept applicable to how systems are connected, and the possible interactions that surge from them.

    - A fine example can be found in *Fire Emblem: Thracia 776* (Kaga, 1999), where there is a direct correlation between captured (instead of killed) enemy troops and weapon availability. Because weapons and money are very scarce, the game builds on the theme of leadership by proposing a softer, more human approach to warfare at the cost of personal responsibility.

    - Player dynamics (Hunicke, LeBlanc, & Zubek) can also reinforce themes, even if they are, most of the time, much more unpredictable to control. In *Pokémon*, training your first party member by switching to a stronger teammate in order to defeat the opponent is a dynamic that consolidates the experience as a caring endeavor for strengthening your pets.

- Mechanical individuality: if a mechanic is available to multiple characters, the evolution and differences between them is a clear framing-based contrast theme, whether there is control of both characters or whether they inhabit the same playthrough space.

- Ludonarrative dissonance: despite the wide problematic it has presented in the field of narrative game design, it is a useful tool to bring attention to contradictions between story and game, those that can then be corrected to enhance certain themes, or kept to create an internal conflict to the player.

- *Spec Ops: The Line* does something similar to this, albeit the dissonance is majorly derived more from the ludic culture developed around military settings, taking on a more meta contextual level through the in-game elements (Payne, 2014).

3.3.3. System theming

Player action and agency, however, are not pushed at the forefront of the media's experience alone[31], and instead, carry a series of system-dependent elements that are also worth discussing to get a complete picture of videogame theming.

- Choices: not as the player decision, but as the conflict presented by the system, which the player action has to resolve. Choices' thematic depth is usually tied to their narrative context: to kill or not to kill, help one or help the other, rebel or submit, speak up or remain quiet… Because the themes are being discussed as part of the system, however, considerations on its traditional narrative potential will not be considered, as they were already covered in 3.3.1. Theme typology. What was found meaningful of choices as a system structure were three particular elements:

  - Pressure: the limitation of time to take informed decisions gains much relevance, because it (tends to) invalidate the figure of the heroic savior, bringing it down to the harsher reality of responsible, visible decision-making. This ability of turning what can usually be seen as a plot device (in the worse cases, plot armor) into a very human moment that most people are not prepared to confront comfortably, regardless of whether player action ends up successfully or not, could become, due to that state of vulnerability, a door to introspection and a more serious reading of the game's themes.

  - Contextualization: while traditional narrative always used contextualization to flesh out conflicts, videogames empower this element even more, because by putting the conflict's resolution on player hands, and laying the information to solve said conflict across a scenario instead of giving it directly to the player,

[31] Mechanics can, after all, be performed by the system too; in a way, the system has a sense of agency

any misjudgment derived from a lack of information is the player's fault. Because it is in their hands to put the effort to reach an ideal resolution, they have to judge the situation and keep acting until they reach a satisfying answer; this process of consideration is, at the end of the day, a thematic introspection. This concept has a close connection with the following concept.

- Availability: the possibility of adding choices in a play space comes with the possibility of limiting them. Availability is not so relevant in terms of solution permutations, but in how designers take agency to allow or prohibit certain decisions. This is closely related with contextualization in the sense that space traversal (or any information acquisition method) can be misleading to thematically point out the metaphorical complexity of the situation. From the three particular properties of choices, this one is the most problematic to implement, because betraying player expectations that have been purposely set up to build on the experience can be easily seen as cheap and lazy.

- Diegetic dynamism: videogame's need, the one to simulate a comprehensible space filled with agents (sentient or not) that can be interacted with, usually implies a degree of environmental change. Such degree is usually conditioned by development costs, but it can be used as a thematic outlet, furthering the relation between the player and the environment through how each changes the other, or how they are prone to change by themselves. This can go from scheduled activities, to entity control from dynamic difficulty adjustment systems, or how the design of a space conditions the player to act, the meaning being on how the environment "tries to control" player action[32].

- Game loops: loops are a firmly established game design structure that is used to define player behavior, usually to represent the general, minimal necessary actions taken to surpass each of the challenge variations that will be presented through the progression of a game. This approach heavily focuses on repeating the essence of the activity in different contexts, which is the exact same behavior that themes use to make themselves noticeable. Having established that mechanics (the most common components of game

[32] While not meaningfully pertinent to the overall thematic discussion, examples of environment conditioning would be things like negative space, focal points, verticality, mechanically-locked spaces, collectables…

loops) can be designed to have thematic relevance, game loops become one of the most powerful ways to express thematic variety, especially in dynamic contexts where the specifics of interactions player – world are not linear.

3.3.4. Thematic noise

After exploring the thematic possibilities of videogames, it would be appropriate to take a look at one of the major drawbacks that their implementation incurs in. Videogames, unlike many other artistic media, require a simulation of a self-sustained, mostly coherent, play space to have the player in. Because the need comes before the artistic intention, and its goal of mirroring reality can easily entail a relative complexity foreign to the story's themes, such prerequisite is not always in line with the artistic potential of the artwork. An abundance of non-artistic components (as in "not emotionally meaningful") in an artwork is what we call thematic noise.

The notion that everything in an artwork must have purpose is a relatively obvious one: if something feels out of place or does not contribute, having it be part of the experience is a detriment to the creator's vision of how it should be. When creating with an artistic view, at least from our point of view, the same reasoning applies under the idea that everything in an artwork must have artistic purpose.

The fundament of the artistic view, however, comes from much deeper, because being able to meaningfully impact someone is not as simple as entertaining. To achieve it, we deem two factors to be, at the highest level, needed: firstly, the emotionally resonant content; secondly, uniqueness. While the degree of uniqueness required to differentiate oneself is a discussable matter, the impact of seeing something (in this case, thematic construction) for the first time amplifies the emotional effect, which is still a relevant part of meaningful emotionality. In summary, having as many game systems as possible reflect the artwork's themes requires an effort, and creates a cognitive complexity, that adds uniqueness to the artwork, making it more probable that it can meaningfully impact someone, thus, more probable that it achieves its artistic purpose[33].

The importance of thematic noise, however, is not how it limits achieving an impossible ideal of structural perfection, but how it is a detriment to the artistic quality of an artwork. As it was

[33] Not to say that the only way of furthering the "artistic" quality is through themes, but it is the purpose of scaling themes, and making sure to extend their influence to as many components of the artwork as possible.

discussed previously, the interpretation of a theme is something rather subjective, and the presence of dissonant elements can confuse, tarnish or even misdirect the interpretation of the author's intent, making it more difficult, then, to achieve meaningful emotionality.

Videogames in specific tend to have very prominent thematic noise-causing structures pre-built in the conventions of the media, mostly because of a system being implemented to solve a gameplay design issue, without considerations for the narrative implications it carries: failure or death, utilitarianism in choices, infinite retries, and even game loops, which were mentioned before as a great way to emphasize themes through organic repetition, but which can also imply ideas about cyclical determinism (Hoefer, 2016)[34].

3.3.5. Themes and rules

Implementing theme control in a game means defining how the game develops narrative through themes, those being defined by a set of rules. Sometimes, this is a fine relation, as a theme aims at communicating something, be it an idea or a feeling, that can be described, and so, represented as a set of rules. A problem surges when that is not the case, what we will call open themes.

Open themes can occupy a myriad of ranges, usually being a specific argument with various points of view that have to interact among them or a sequence of only partially related ideas. The matter in hand is that trying to define certain relations through a set of rules flattens the depth of the thematic discussion, because in some cases, their essence lays precisely in the impossibility of defining them as truths. This is especially true when the authors are not trying to make a specific judgment of value, but instead, a comment on observed human contradiction.

A good example, and one of the most prominent ones in story-telling, are moral choices. Implementing them is relatively easy when they are presented under a defined position, that serves as an overarching theme discussion; it is not the case if they are to be presented more neutrally, when the objective is exploring the player through their own values. In wanting to give designers the maximum freedom to approach their game's narrative, it was then assumed that two types of theme-defining structures would need to be available.

[34] This, on the other hand, can also be used to argue against our existence being limited to repeating a cycle of mundane life until death, by for example, making a game that is about successfully escaping such loops.

3.4. Art-centric criteria's secondary elements

With a developed notion of the project's narrative vision within the videogame media, there are only the final considerations to be made before discussing the system's possible implementations.

While the focus has been set on themes, there are other aspects that contribute to the general artistic value of an artwork. Since we develop a tool to be used in other projects, it is not in our hands to provide all the artistic backbone of the hypothetic game, thus only the concepts considered most relevant will be taken into consideration: emotionality (when building the story), interpretability (degree of theme specificity) and technical style (allow a sense of self, permit stories to be made with a distinct, shared flavor).

Those, together with what has been previously discussed will conform the mindset with which decisions, regarding the entire system design process, will be taken.

3.5 Narrative diversion

To have a better grasp of the possible narrative structures to be implemented, research was conducted to relate choice and narrative progression in interactive media.

At the highest level, a typology was stablished by Pedro Cardoso and Miguel Carvalhais to define player interaction with the narrative system, focusing on the consequences of the choice from the point of view of the player (Cardoso & Carvalhais, 2013):

- Branching: making a choice between mutually excluding paths; generator of points of no return, which in turns means they are limiting factors for plot discovery.

- Bending: decisions taken to expand the narrative playing experience beyond what is mandatory or necessary; it prioritizes engagement and exploration rather than plot continuity.

- Modulating: acting to change the state of the world, with a strong (but not necessary) emphasis on social interaction, altering the relationships between factions or individuals.

- Exploiting: acting outside of what the system had planned, resulting in the usage of (intentionally or not) exploits, glitches…

For a system that regulates interactive dynamic narrative, all of those types of choice are relevant, since it would be needed to define what the player can and cannot undo, to give dimensions to the world and their characters, to allow for non-plot defining meaningful interaction, and to prepare the system so that it stays stable even if its limits can be occasionally trespassed.

Another typology can be presented from the perspective of traversing the game's narrative structure, in this case, based on the plot structures of the "choose your own adventure games" genre (Ashwell, 2015):

- Time cave: the story constantly diverges, having shorter length in exchange for more varied replayability.

- Gauntlet: the plot has a linear, clearly defined path that can diverge, only to return to the main story, be it through progression, failure or return to previous beats.

- Branch and bottleneck: through the constant tracking of story states, the narrative diverges into multiple choices at once, which are eventually rejoined into a linear path again, before branching anew.

- Quest: the narrative sustains diverging into many paths, each having a cluster of nodes with multiple connections (conforming a sub-narrative within, usually identified as a "quest"), each cluster rejoining at some point, but keeping said groups alienated from each other as independent narratives.

- Open map: there is no order to the node structure, each possibly connecting with many others, keeping track of progress through states, and with the possibility of visiting one node in multiple ways.

- Floating modules: the story has no central connection and instead is composed of multiple self-contained beats forming a sequence of causally unrelated events presented as a whole.

- Loop and grow: the central narrative constantly loops, and any diverging choices are available, or not, depending on the registered player states, soon returning to the main structure until the state for ending the loop (if it exists) is reached.

The system of this research is catered towards an "open map" structure, a world fully controlled by states that allows complete player agency to go back and forth, modelling the narrative most suited to each player's actions. However, this typology is especially useful, in the form of templates, to consider the structures needed in case it was desired to make a specific type of narrative out of a character or a specific world state (for example, having a "loop and grow" narrative for an NPC, the content of which could still be defined (or not) by the parametrized states of each agent, but which provided a default structure that facilitated thematic flair).

3.6. Observations on moral choices

Similarly to the problematic of making rules for themes, a consideration must be done on what factors affect moral choices, or the aspects that could be read as ethic in a given context. Inspired by Failbetter Games' typology of moral choices, some aspects were taken for consideration (FailbetterGames, 2012):

- When does a moral choice become a moral choice to be commented on: there can be situations where the player makes a decision that has moral connotations; since they are being taken within the narrative world, they are part of the narrative, but it does not mean it is built into its dynamically constructed themes. Easy solutions would be to ensure all meaningful choices are encapsulated within the desired narrative, or that a system makes sure no moral decisions can exist outside of it. It is also possible to only take into consideration the choices relevant to the plot, but such approach could incur into ludonarrative dissonance.

- Pressure-based context: if judgement is passed based on the player available options, there will be a need to consider context, as, under some circumstances, not all actions can be as reprehensible as they would in other situations, thus should not penalize equally.

- Social-based context: because the objective is to hold a conversation between the game's systems and the player, it is not sensible to establish a rigid moral standard. However, if we judge based on the NPCs' criteria, we have to consider biased elements, like who was affected by the moral conflict, how they were affected by its culmination, how was the player regarded before it (if there was a positive relationship, it is easier to be forgiven or receive a lesser penalization, while having a negative relationship can easily scale previous confrontational elements), whether external agents could know if the player was involved (and how it could be proved), how different individuals in one same collective react to it…

- Rewarding: if the nature of moral choices lays in the choice itself, adding a reward to them invalidates their primal purpose; even if they do not provide a ludic reward, the consequences of the narrative should be enough. The problem surges when we are including rewards to sway player towards a certain, not necessarily clear, "evil" moral choice, usually to remark ideas of temptation, compromise, moral ambiguity, etc. Which, of course are not themes desirable in every story. A sub-system that created rewards for certain choices could be implemented with the option to toggle between enabled and disabled; alternatively, they could be hidden until the choice is taken. In either case, it is unlikely rewards will be implemented in our system to begin with (due to time constrains), but the consideration was worth noting.

- Available information: since moral decisions are catered towards meaningful, personal insight, the choice should be, without ruining narrative tension, consciously taken with near-perfect pre-effect information. This could end up not being the case, but any attempt to build from a deceitful set-up can hinder the narrative's effectiveness. While the acquisition of knowledge can be preemptively regulated by the player's actions in

the world[35], thus never feeling like it is a developers' oversight, it seems wise to account for a way to control that choices do not scale to the point of being mostly incomprehensible to the player.

3.7. The case for procedural narrative

In the current landscape of narrative generation, procedural narrative is vaguely defined as story-based authored content randomly selected, usually after being brought out by player action (McRae, s.f.)[36]. Be it linear of fragmented, it takes pre-defined pieces and assembles them together to form a whole. The reason why it was not mentioned beyond this segment of the discussion is because this is a design approach completely contrary to the one established at the beginning of this paper.

The idea behind procedural narrative is to make bits of stories, and then present them in a pseudo-random order as the components for a plot. Sometimes, the criteria that chooses said content is totally random or only tangentially related to player action, and other times, the procedural treatment is used to define character traits, player goals… before the story begins, as the basis for any other narrative controlling system. In any of those cases, procedural narratives are usually applied so that games can provide more hours of content without lacking new narrative content, or in order to provide a unique experience on a player basis, while having strict authorial control.

Essentially, it tends to share the same problems as linear narratives, as it can hinder player agency through the imposition of authored content (although it can consider player input between each bit, which has its own advantages) and its authoring burden still exists, but rejecting the strongest attribute of linear storytelling, which is the capacity to control story progress, delivery and cohesion. This is all in favor of novelty, contrary to my approach, which favors meaningful intent above all.

In conclusion, it offers a narrative model that does not reach the possibilities and desired depth that emergent narrative does.

[35] Being a dynamically composed, simulated narrative, there are no pre-planned requirements to take a decision, thus acting is mostly an intrinsic player action, not an imposed one.
[36] Procedural narrative as in a narrative created by an AI is possible nowadays, but the quality standard is so low that it is not worth mentioning.

3.8. Summary of the theoretical framework

When constructing a structure-defining narrative system that has no pre-determined narrative to adjust to, generalist measures have to be taken to ensure that as many scenarios as possible can fit the mold.

In order to keep an artistic consistency that enhances the final artwork, an artistic approach has to be taken. For that, the most meaningful story elements, themes, would be made into the system's core. Incidentally, the narrative universality of themes would allow the system to reconcile mechanics and story, furthering both coherence and player expression through the usage of emergent narrative, achieving a whole catered towards naturally generating communicative sequences between game and player, with information being delivered in both directions.

4. Cases of study

The aim of all cases of study was to analyze how dynamic interaction systems affected the game's narrative. What this meant was that focus would be put on the elements that were considered to provide value, by letting the player develop organic narrative paths, that is, the opposite of scripted or bottlenecked events. Heavy focus is put on the specific interactions that make the game, instead of the entire system, because it was deemed better to provide a general framework that included all possibilities, rather than designing a single, focused system for a specific type of play.

It is worth noting, however, that we did not want to make a hard selection, since our objective was to create a system that allowed designing a wide range of different narrative progression, thus if anything seemed worth noting, even if it was only tangentially related, it would be considered valuable and at least mentioned in each case of study.

As it has been a constant up until now, a disclaimer has to be made: the cases of study were a source of information, explained to justify the conclusions (at the Technical demo section) that brought the research to certain design decisions, but they do not contain any direct system justification.

4.1. Fallout: New Vegas

Before we enter the specifics of *Fallout: New Vegas*' main interest point, the quest system, there were some minor but important observations to be done on the main systems that interact with the quest's one.

Firstly, we touch upon the two main numerical checks used to progress questlines: skills (which are upgradable as the player levels up) and primary stats (a set of numbers chosen by the player at the beginning of the game, and which remain mostly static for the whole playthrough).

The way in which such elements interact with narrative progression is through said numerical checks, skipping or progressing quests if the player has a big enough number, while hindering or even cutting it off if the check is not passed. The most relevant observation here, to our case, is that any system that reflects this kind of pattern works better when those checks are not abused as arbitrary content lockers, and that there should be secondary, more general ways to progress quests, albeit adding a sort of penalty that retains the value of passing that check (be

in invested time, narrative results, progress conditions…). This approach is clearly the most desired to implement in our case, because it firmly points towards having a system that accounts for multiple progression conditions at once, with the possibility for different outcomes depending on which one is met, allowing for a thematic spectrum to be represented.

Another relevant system is the companion one, through which the player can be accompanied by certain NPCs with usually other quests tied to them. What is most interesting to note is that, by spending time with them, new content is unlocked, and the conversations that lead to this process have priority over others to make raise the possibilities of the player stumbling upon it. While it was unknown, early on, whether we would have to interact with a dialogue system, it was at least clear the versatility of event priority when more than one was available (Fallout: New Vegas Companions, s.f.).

Next is the karma system, which is supposed to be the moral compass of the game, and one of the two ways in which NPC affinities are decided. Interesting observations on it are, for example, that depending on who you kill (and not on the action itself) you get good or bad karma, which implied it may be preferable to allow checks within specific narrative conditions instead of stablishing very general rules tied to player actions, but detached from context.

Another interesting interaction with this system is related to some companions, that stop following and aiding the player if they get too much bad karma. While this only happens if the player speaks with them, we see much more narrative potential on triggering this event automatically (although not necessarily at the exact moment in which the player crosses the threshold), which would require agents to be able to receive game events, check if they are fulfilled, and then act correspondently (Fallout: New Vegas Karma, s.f.).

The last system we will mention is the reputation system, which defines general faction affinity to the player character (Fallout: New Vegas Reputation, s.f.). Its most notable aspect is that, unlike the karma system, it works in two axes, a fame and an infamy one, which in conjunction form the final result. This means one can have very good and bad affinity, and be considered a very polarizing figure within the community, or be very neutral, or in one of the extremes, providing more depth and diversity to set certain scenarios up.

Within this system, there are many noteworthy considerations:

- Actions only register in the group’s reputation if they are observed or if it makes sense that they could be attributed to the player, meaning there might be a need for a check of

situations that have to be validated to alter reputation values. As an extra consideration, while we did not find any case in this game, it would be interesting to adapt the system in any way necessary to allow the developers to add NPC actions that could be attributed to the player even if they did not do them, and maybe even allowing them to prove they were not responsible for it, and reversing the reputation alteration[37].

- Reputation applies to factions and locations, and while it is kept separate in this game, we should account for the possibility to have characters affiliated to multiple factions, be it by separating them by cluster, of by making hierarchized faction layers.

- While there are very few instances of this, sometimes a whole faction's reputation can be reset. It would help a lot with believability that there was a personal reputation separate from the faction one, so that an antagonized NPC does not suddenly behave friendly, or vice versa. It would also be interesting if we could add consequences for an NPC acting on personal behalf, and against their faction's orders.

- Reputation gain is immediate, sometimes deriving in NPCs attacking the player out of the blue because an action that does not seem to imply infamy, but which does give antagonizing points, is performed. A solution to this could be postposed reactions, to simulate the news being shared instead of everyone being part of a hive mind.

- Sometimes, reputation is used to determine a speech check. Consequently, the system should check beyond the player stats to devise narrative progression (although without the prior knowledge of how is the system, this consideration is only serviceable as the idea that narrative progression should be multifaceted, and not limited to a single cluster of information).

- Many times, free resources can be earned by interacting with a faction with which the player has a good reputation, and doing so does not affect the reputation value in any way. It was interesting then to consider how those "free rewards" (although it does not have to be limited to material gain, it could also relate to social relations, although then

[37] Note how this approach is only advised if the NPC action is a consequence of a previous player choice, otherwise we are affecting the player's situation without a meaningful purpose.

it should be considered case by case) could have a boomerang effect if the player switches sides at some point, meaning the more advantage you have taken of a faction, the more they will resent the player if it betrays them.

On a quest by quest basis, the most enlightening insight was described in the following list (Fallout: New Vegas quests, n.d.) and (TheSaltFactory, 2020)):

- The game has a myriad of mission where there is a very specific logic being used, which is founded on agent knowledge, instead of doing so at a system level, making it so points of no return are not pre-scripted, and instead happen organically and naturally, regardless of the meta-context. An example of this would be cases of false allegiances, or how some companions from pre-existing factions exhibit different behaviors to the same player – faction relations, based on seeing (or any other sensorial process pertinent) the player do the actions that define the faction's posture or not.

- Taking certain actions implies the progression of different quests regardless of how involved the player was with said quest. This approach implies that they should not be serialized, linear instructions, but be stored as condition checks, active even if the player has technically not interacted with the first sequence yet, to fill the world with a sense of logical interconnectivity.

- Depending on the player's affinities, some challenges vary their content, meaning it can influence narrative contexts both accumulatively and on a binary basis (for example, amount of enemies, and / or from which factions they are). Whatever approach we end up taking, we should be able to translate multiple conditioned checking to a single event trigger.

- An NPC's behavior being affected by another NPC might change personal allegiances and fail certain quests[38]. To regulate such behaviors only when desired, a very solid affinity system had to be made, instead of fully depending on player actions triggering events that alter the correspondent reputation value, tied to arbitrary quantitative checks.

[38] While we are not fond of quest lists, which can both trivialize the meaning of a quest, and make it more of a chore, if such approach was to be taken, a lot of care should be put in giving the proper feedback on what a player can do, and what they have been locked of as a consequence of NPC action.

- To avoid instances in which affinity and actions do not align, as it is the case with soldiers from some factions not reacting to their own troops' being left wasted in the desert, it would be appreciated to have a debugging system that covers when affinity-related interactions such occur, but no response is given.

4.2. The Elder Scrolls V: Skyrim

Skyrim (Howard, Pagliarulo, Nesmith, & Kuhlmann, 2011) is an open-world game that uses the same Radiant AI system base as *Fallout: New Vegas* to control NPC behavior. The use that it was given in this game peeked our interest, and prompted us to make further observations on it (The Elder Scrolls: Skyrim Radiant A.I., s.f.):

- Under preset circumstances, a quest package can be used to generate a whole mission in an unexplored place, with difficulty adapted enemies that can have pre-world state considerations built in their narratives. While many of those features enter more the realm of procedural generation, the idea of unlocking chunks of content through single, dynamic events, and to do so to support certain design pillars (in this case, improve exploration) was considered relevant to the research.

- NPC to NPC dialogue interaction is mentioned to occur, specifically to comment on the player's reputation and actions: tools should be provided to allow a wide range of inter-system communication.

- The system, although this seems more like an AI scripted behavior, also reacts to more general actions (in the sense that they are not delivered upon an agent), like having an item dropped in front of them, which made us consider how we can adapt generic, non-parametrized actions as readable ones to the narrative manager.

4.3. Dark Souls Series

The *Darks Souls* series is, strictly speaking, comprised of three games with very similar mechanics and systems, which is why they will be discussed all together (FromSoftware, 2018), (FromSoftware, 2015) and (FromSoftware, 2016).

The reason why this game gets as much attention in this research as it does is the freedom it provides when dealing with quests, despite how relatively simple most of them are, especially due to the focus of integrating them in the diegesis, instead of guiding the player through them with extra-diegetic elements.

A technique they constantly employ, especially in the first and third game, is altering the position of the NPCs along the quest's progression, usually after a condition is met, or dialogue with them is exhausted. Seeing how effectively this helps to keep an organic feeling of independency, as well as accompanying the player across their journey, it would be wise to account for internal events on top of the systems that control reactions caused by the player and by other entities' presence. In essence this would mean having events that may modify character context but which are purely decided through developer standards, not controlled by the player's actions, or that the world control system can alter (and is incentivized to change) agent position.

While also desirable that such events are controlled automatically (when relevant), there would be a level of complexity to pay attention to in the form of controlling internal events that alter affinity: in the first game, after certain progress is made, a progression-relevant NPC gets killed, limiting respawn spots for the player. If we took this context to a game with dynamic narrative systems, it would mean, taking the *Fallout* approach, that we would be generating a reputation loss from the victim's faction towards the aggressor's. In a hypothetical not concerning *Dark Souls*, then, we could have two new different characters (from opposite factions) that could be both beneficial to the player, suddenly becoming enemies and killing each other because of this very specific incident at a completely different place.

Again, this relates to the idea of how updates of reputation extend, and it is very possible that a proper subsystem needs to be developed to avoid anti-natural cases like this. Another idea to derive from this example, and one that *Dark Souls II* pioneered in the franchise, is that we may want the player to witness the action, in which case we would have to be able to manage scripted events too.

Incidentally, and although this may be stepping too much into AI behavior design, it would be desirable to consider what would happen if an AI was given the command to attack the player for their hostile affinity, and then found another NPC whom they are also opposed with. The reason why this is relevant is because it may affect the reputation system, adding conditions of priority: some characters may act more in the spur of the moment and keep their eyes on the first target, others may prefer to back off when they are outnumbered, some may go after the one who belongs to the faction theirs' oppose instead of dealing with the personal affairs… All of those considerations were, at the time, judged necessary to be provided by the dynamic narrative manager, not specified case by case.

Jumping into the various quests of the series, the following is a list of the most relevant insight:

- Many events, while pre-scripted, require multiple conversations to simulate time, an effect that could be accomplished by actually having timer-based events[39].

- Some characters' affinity can increase so much that they stop basic behaviors like retaliating after being attacked. This suggests that each affinity level of each faction has its own general response, which then will have to merge with personal behaviors, meaning we can delegate huge chunks of narrative management to behavior blocks external from our system, highly alleviating the workload.

- With so much dynamic character movement, we were compelled to question ourselves how would we manage the difference between the event of "the character progresses to the next quest stage" and "the character has already got there". If the designer wants to give that responsibility to the narrative manager system, we should be able to respond properly.

- Some characters exhibit personalities that are reversed to trap the player. This kind of behavior, in which polarized actions are taken in neutral affinity, could be scripted, but more interestingly, serialized and predetermined by extending the behavior block approach to a personal basis. The rawest implementation we could think of was having a general index of actions, then specifying both for faction and personal blocks which are considered appropriate in each state.

[39] Although in the case *Dark Souls*, it was probably implemented this way to find a fine compromise between player convenience and narrative drama.

- Certain unique objects can be obtained differently depending on how the player interacts and progresses the quests they are involved in, altering the methods, but not the rewards that completing them give. This made us think about how we could add a subsystem that supported narrative progression by taking an object and a list of the possible scenarios in which it could be dropped, serializing the method instead of coding it at every instance.

- In some instances, the game gives the option to the player to lie to NPCs. This made us think about how we determine when we allow the player to lie (ideally, we should always be able to give them that option), and how do we manage (at first glance it should be by personality type, although context could be important) each lie for each character, especially if it's added as a generic action, not a specific choice. Again, such a minor inconvenience may go out of scope, but this was left to be considered at the technical demo's planning.

4.4. Façade

Façade was an academic project that aimed at advancing the notion of high actor agency experiences, including both the player and game entities, and specifically focusing on a narrative driven by the characters' dynamic behavior.

In essence, this project is taking the traditional branching narrative approach and altering both the input options, expanding player agency tremendously, and later having to classify that free-ranged input into similar predefined categories that most branching narratives try to limitedly simulate. Not only it adapts the state of the game to any written player input, most nodes of the narrative are independent from previous ones, better fitting the procedural qualities of the project. In summary, it is a designed story-telling format that can be successful in achieving dramatic effect through various combinations of the same pre-defined set pieces.

This highly contrasted with the aims of our research, which tried to define a system that controlled the state of a world and the interactions within, trading specificity for serializable and diverse elements that adapt to many design contexts. Moreover, we wanted to achieve this

through diversity of action, while *Façade* focuses very prominently and uniquely on free player discourse (with the exception of extremely limited mechanical actions like hugging or kissing).

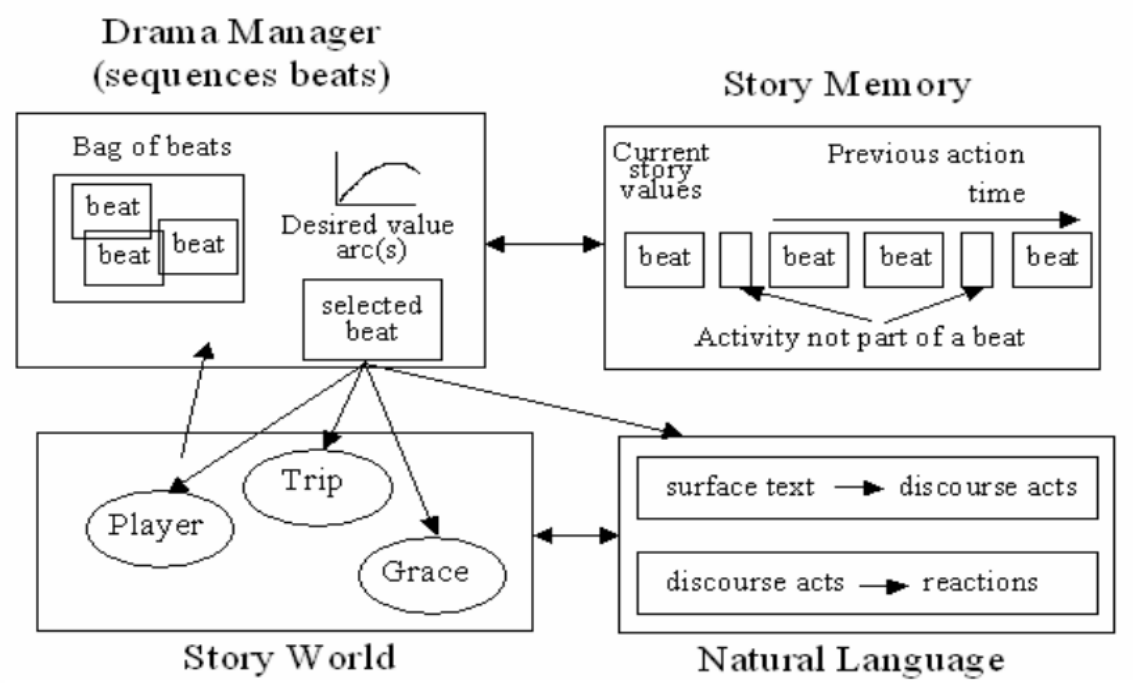
In regards of serializable methodologies, the authors themselves mentioned that the workload was incredibly taxing, despite limiting their narrative to two characters, a single common space and playtime of ~20 minutes. Said best in their words, "*Façade* is not generating sentences – although it is generating sequences" (Mateas & Stern, 2005, pág. 5). The authors proposed two solutions to speed up the development process. On one hand, narrative specifications could be abstracted at a higher level, something we were already striving towards, in the form of applying generally identical behavior blocks with character specific alterations. On the other hand, they proposed low-level improvements, like procedural animations or automated non-verbal gestures that could be organically reused instead of constantly adding slightly different dialogue to nail the transitions between narrative beats. This second proposal may be less applicable in our case, since we did not expect to engage with dialogue specifics, but was still taken into considerations when setting the system's goals.

Before going onto specifics of how the project was implemented, a noteworthy observation was found in their "Failures and Successes" section (Mateas & Stern, 2005, pág. 5), where they noted how they had trouble communicating the personal states of the characters through their interactions, which is exactly something we had risen concern about previously, as we are trying to guide decision-making through social contexts, which both need a grasp of the situation and making the right interpretations, two factors that will hardly be universal to all players.

4.4.1. System overview

To understand the reasoning behind the system's implementation, we have to look at how the authors described the general approaches to non-linear narratives they knew of (Mateas & Stern, 2003): the first was to hand-craft a structure of nodes, each being a chunk of content connected to other nodes, which the player would advance through until they reached a conclusion; the second was to create a procedurally generated environment, with each agent having assigned a certain behavior, but completely ignoring any overall structure, focusing only on the chains of action-reaction between elements, then letting the player act upon such a world. Theirs was an attempt to capture the structured pacing of the first approach with the agency and freedom of the second, which would translate into autonomous actors that were directed at specific points by a more general drama manager.

Figure 8 Façade's internal structure



A visual representation of Façade's internal structure, made by the authors (Mateas & Stern, 2003)

4.4.2. Natural Language Processing (NPL)

Natural language processing (NPL) of player input is the system responsible of interpreting the sentiment inherent to the player input, identifying keywords tied to pre-defined clusters of feelings, then returning a response based on how that feeling plays into the context defined in the drama manager (Mateas & Stern, 2004). Despite the actions of the NPL usually creating a reactive story beat that substitutes the previous one, it is a system we have mostly dismissed, as it focuses very specifically into analyzing language-based player input.

4.4.3. Drama manager

Much more relevant to our case is the drama manager / beat sequencer (Mateas & Stern, 2005). This is the system responsible to build the story as it goes on, dividing actions mainly in two sub-games the player partakes in:

- Affinity game: actions that define whose side the player is on, affecting how the agents respond to the player, and which routes they have available to progress through.

- Therapy game: actions that allow the agents, based on their context(s), to acknowledge their problems, defining the stage of their conflict, and how close they are to certain conflict resolutions.

The evaluation of both an agent's internal state and relations to others is done by assigning values to each of those aspects, called "social scores", which will adapt depending on the actions and contexts of the rest of the participants. To better define what contexts are, we took the liberty to cite the authors directly (Mateas & Stern, 2003):

> Each beat is designed to be able to respond to any of these discourse acts at any time, in a manner appropriate to its currently active context(s). A context is implemented as a set of forward-chaining mapping rules defining how to react to player action. The current beat activates multiple individual contexts, typically a single unique custom context and one more global contexts, each at their own priority. Global contexts tend to be reused among beats, so an author usually only has to create one new unique context per beat. (p. 7)

In terms of building blocks, the drama manager has as its basic unit the story beat, with only one active at a time. Each is comprised of a series of instructions that lead to a goal, and a continuation after an endpoint is reached (be it another predefined beat or a request to generate a new one), whether the goal was achieved or not, followed by a transition goal that serves as a bridge between both beats. To be considered a beat, it's successful conclusion must alter in some way the social scores. Most content is enclosed in those predefined beats, but some filler was added to cover for player input that could not be accounted for, thus not affecting story progression, but at least having a response to it before returning to the current beat, or skipping to another one.

A more generalist structure is the global mix-ins, which codify a specific auto-conclusive topic with multiple layers of beats, defining more high-level parts of the story. They are entered or returned to from any point the drama manager sees fit, or left to possibly come back to it at the appropriate moment, as well. Each of the beats that compose them has a gist point that determines where the conversation should be returned to, and also whether a particular beat can be determined as conclusively explored or not.

The exact process through which relevant story progression decisions are taken is based on a series of variables common to all beats (Mateas & Stern, 2003, pág. 10):

- Preconditions: checks that define whether a beat is available or not.
- Weights: values that define the probability of a beat being chosen, within a range of 0 to 1.
- Weight tests: checks that can alter the base weight of a beat.
- Priority: value that established the tier of importance of a beat.
- Priority tests: checks that can alter the priority of a beat.
- Effects: changes that will be applied to the social scores in case the beat is successfully traversed (past the gist point).

In consideration of those, the process follows as such:

1. Evaluation of relevant beat-specific states to compute "Satisfied" set.
2. Evaluation of priority to compute "HighestPriority" set.
3. Score each beat based on how satisfying their narrative effect would be, to compute "ScoredHighestPriority" set.
4. Multiplication of each score by the beat's weight to compute "WeightedScoredHighestPriority" set.
5. Randomly select a beat based on the probability defined distribution.

The variable that allows to define which beats are adequate based on the context is the tension level, controlled by the player's moves, to adjust the story flow to an ideal of Aristotelian tension arc.

4.4.4. A Behavior Language (ABL)

A Behavior Language (ABL) is, to put it in simple terms, the agent director, breaking down the instructions of each story beat, and taking care of the moment to moment decision-making of each actor (Mateas & Stern, A Behavior Language for Story-based Believable Agents, 2002). It is responsible for their position, movement, physical interactions, reactions, dialogues and the storage of Working Memory Elements (WME), which basically serve to keep track of the events to avoid repeating them if it would not make sense to do so, with the appropriate responses in case illogical behavior was being enacted by the player (for example, it would not make sense for an actor to respond in the same way to the player knocking on the door when they arrive than if they do it after the door has already been open).

Each behavior (subdivisions of each beat) has a series of instructions, and when one of them is not fulfilled (failed), that particular behavior stops and is suspended. This clarification is done because many behaviors can be playing at the same time, some even being recursive (for example, checking the time every once in a while). There are four basic types of instructions:

- Wait: keeps that agent's behavior frozen, but does not affect other behaviors that may be playing at the same time.
- Act: defines physical responses.
- Subgoal: a call to another behavior. This can cause a case of recursion, and it can complicate beat management when new instructions come from the drama manager, and multiple behaviors have to be cancelled at the same time.
- Mental acts: internal calculations that adjust the game state to properly simulate the story.

Since behaviors are functions, they can englobe various versions through the use of polymorphic functions to adjust the response to the context, using a single behavior call. *Façade* manages this by executing all such functions, and choosing the one which can have all of its instructions succeed. If there were multiple behaviors that could be completed, a specificity value is checked as a priority pass; if they were to coincide in this aspect too, one would be chosen randomly. If a behavior returns failure, and there's another behavior available for that same polymorphic call, it will enact the second one as long as it can be successful. The system also allows and prevents in-behavior interruption, as well as priority or other property modification, depending on how the behavior is received and which behavior locks have been added (Mateas & Stern, 2002, pp. 3-5).

To support multiple and coordinated agent activity, functions named "joint actions" were used. Those tell the agent manager who are the participant actors on that behavior, and makes sure to notify any internal entering or exiting from it to the rest, in order to stay coordinated. Synchronization is achieved by a two-phased protocol: the initiating agent broadcasts an intention to the rest; those respond with their own intention, whether they will succeed, fail, enter or not enter a certain subgoal…; then the initial agent performs or alters their behavior accordingly. This coordination is handled by creating a team WME that is accessible for all the agents. The waiting of success or failure confirmations to continue to the next goal, again, does not impede the execution of certain behaviors (for example, body language or position changes)

that may be desirable to autonomously execute in the meantime. Once the bottleneck event's results are confirmed, all agents receive their next task for that point in the event.

To deal with player action interrupting such behaviors, multiple elements are used (Mateas & Stern, 2002, p. 9).

- Beat goals: each instruction that conforms each beat, and which allow an interruption of the beat's progression between, but not during, them.
- Handlers: code units that set a priority to a player action, and decide if they are more important to deal with than the current behavior; they fail it if necessary (and possibly re-run such behavior adequately to the context), and depending on the stage in which it was interrupted, it yields different predefined results.
- Cross-beat behaviors: beat goals that can be interrupted in their performance (usually longer but single actions, like a dialogue line or a subgoal).

4.5. F.E.A.R

F.E.A.R is a survival horror, first person shooter that controlled its AI based on agent decision-making instead of long strings of fixed states (Orkin, 2006), of which only three are used: Goto (to move a character), Animate (to change a character's animation) and UseSmartObject (an internal indicator of the animation that Animate has to use).

The philosophy behind this is that every state change, is, in reality, a position movement and / or an animation change. To decide which movement and animations to choose next, a planning system was implemented: this is based on defining the goals and available actions, and letting the AI decide how to reach them, instead of defining each state's transition with a traditional finite state machine (FSM).

Planning is defined as the formalized process of searching for sequences of actions to satisfy a goal, in the case of *F.E.A.*R, resembling the STRIPS planning system. In planning systems, the goals describe a desired world state, while actions are composed of both preconditions, which have to be fulfilled to act, and effects, which defines how the world is influenced.

4.5.1. Planning systems components

State can be described such as (var1, var2, var3), while the world state goal as (value1, value2, value3), with each variable of the state corresponding to the world value it has to match. The agent's goals are competing through their priority and whether they can be attempted or not (regardless of output success[40]).

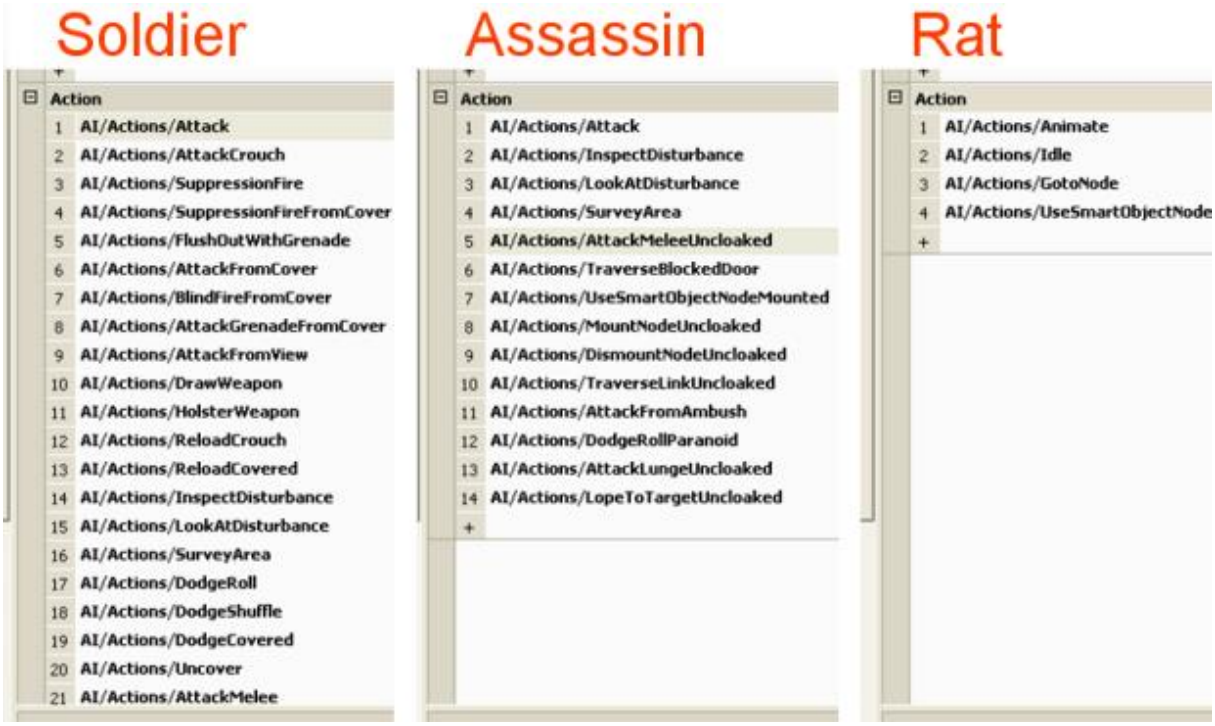
Each time an action is performed, and its effects have to be applied, the variable the effect refers to is removed from the world, and then re-added with the value provided by that action. This is done because in formal logic, nothing impedes a variable from having multiple values, but it is most likely that this possibility will not have to be considered, or at least will be easily controlled, since we will not be working with STRIPS, but with C++.

What defines whether the agent can attempt an action or not depends on whether it is stored in the list of actions they have, called an action set. Every time an action, which was reported to be a C++ class object, is added to an agent, it gets initialized with the adequate values, which permits different goal responses with the same sets of actions for different agents.

Figure 9 F.E.A.R's Action Set example

made by the authors (Orkin, 2006)

[40] This is relatively dependent, but the gist of it is that a game planning system is not after efficiency, but human-like decisions



4.5.2. Additions to the planning system

The developers of *F.E.A.R* built on the STRIPS system to accommodate their implementation, as they only based their design around it, with the following changes:

- Adding a cost per action to have prioritization, forming a nodal structure that is traversed with an A* algorithm[41], where the nodes are states of the world, the edges being weighted actions, and the objective being to reach from the current world state to the goal world state.
- Instead of removing the add and delete lists that control world state, it is done through a fixed sized array that facilitates finding information, at the cost of having to implement sub-systems that allowed multiple element reasoning (be it multiple weapons, targets…).
- Adding procedural preconditions, which are generated when an action that contains one is being considered (for example, those related with mesh traversal, which depends on whether a certain path is available or not at the moment); and procedural effects, which are activated not when the action is executed, but when it is completed, to avoid instantaneous world state alterations.

[41] One of the most common algorithms to traverse weighted nodes.

Beyond the agent level, a high-level system to control group behavior was implemented. It works by collecting all the AI subjects that have to be guided, and giving them a goal with its respective priority; it can be expected that the agents will follow said goal, but there can still be cases where other goals take priority (like surviving). Because the manager of group behavior checks for the various agent's states, it can send different goals to different actors in response to a single world state (move behind a cover or provide coverage fire, coordinating multiple agents depending on their situation).

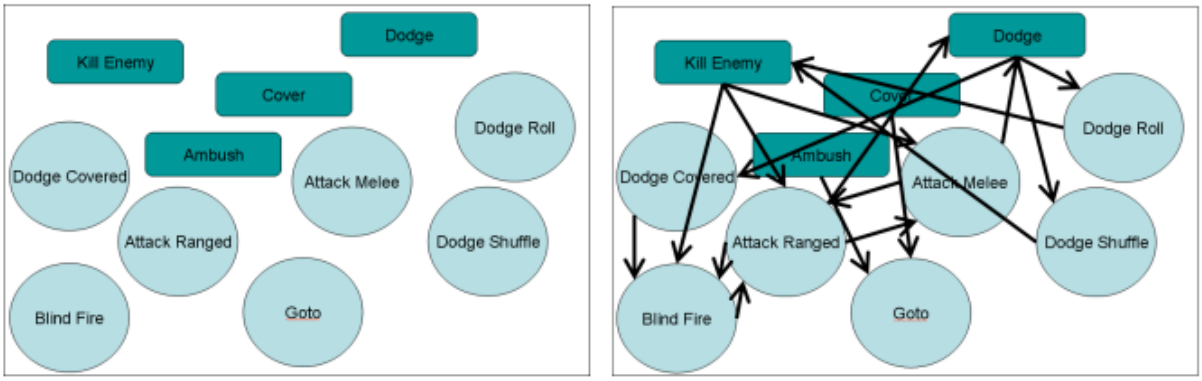
Other usages that can be given to a planning system come from the inspiration for FEAR's implementation (Evans, 2002). On top of what was already explored there, it also mentions the possibility of commanding a behavior, and not only letting the actor decide whether it takes place or not, but to add consequences to disobeying the command; this way, we can add the semblance of moral posture, since there is a choice being made between a penalty and a reward, instead of just an acting decision. They also propose making activities (the goals) external to agents; while this will, again, depend on implementation, the idea is interesting to consider for shared goals, in which we want a global connectivity instead of instantiating the state in each actor.

4.5.3. Advantages of planning systems

This approach was reported to allow decoupling of actions and goals, which kept code simple, as any specific behavior was confined to a specific character, and allowed a general system to alter character states, instead of being uncommunicated from the rest of game actors. It was also argued to facilitate layering behaviors, since the transitions between states do not have to be defined, they are automatically figured out.

Figure 10 F.E.A.R's interaction schema

made by the authors (Orkin, 2006)



Finally, this system was useful to have organic dynamic behavior, as any failed attempt would discard the action in favor of another approach, until all options were used up.

4.6. Degrees of Lewdity

This relatively unknown indie text-based adventure game's study resulted in many fresh and interesting observations to apply to our project.

Something that stood out, for worse, was the need for clauses that disabled already seen events, as experiencing firsthand the pain of going through the same procedurally generated story beats until something new appeared made it clear how detrimental it can be to a game experience (even more if we plan to do one with longer time to go through each beat).

On the positives, there was observed to be a system that modified certain statistics (like level of fitness or study) based on the progress that the player made on the previous days, sort of simulating learning by repetition, thus conditioning player action towards long term goals; if said activities were not performed, the augmented percentage would go negative.

Even more interestingly was how it managed narrative information, which we divided into three sections:

- Plotline presentation: while sometimes the player randomly finds an occurrence that gets them in contact with a web of characters (for example, if things are going too well for the main character, the game could throw a wrench to create tension[42]), many times there simply are characters in the accessible areas, which can be interacted in isolation until a plotline is unlocked, at which point the whole ecosystem of that plotline becomes accessible. This way, the game dynamically and organically controls pacing, and

[42] This is an example, no evidence was found of the game doing this, as no documentation was found.

because there is a need to keep the avatar constantly healthy, the player is conditioned against unlocking too many things at the same time, as it would not be possible to keep track of all of it. Moreover, sometimes a bait is sent, to make sure the player does not miss on certain content, but no plotline is forced until the player shows the interest to go check it out, which improves the player experience by both giving agency, and tackling plotlines that have a natural interest for the player.

- Information presentation: there are bits of information that, despite being fixed, are presented differently based on player context, not forcing straight, pragmatic solutions to the presented problem. An example for this is when the character gets infested with parasites, a problem they do not have the information to deal with; nonetheless, the small loop of finding out what is going on until the medicine is found can be skipped if the player puts up with them until they are gone. This self-regulation on problem-tackling helps strengthening the sense of agency too, and stablishes interesting situations of risk-reward. In conclusion, our system should account for monitoring the avatar's context / awareness of their situation, while keeping action-based conditions in case the player figures out a way to proceed by themselves.

- Progress sequencing: the different types of relationships the avatar and an NPC can maintain are not developed with the same actions through different points in time, even if the type of relationships is the same. In other words, the game not only keeps track of different possible types of relationships that can be developed at the same time with a single character (a very interesting way to define human relationships), but said types depend on a limited pool of available actions and contexts. The most common example can be found when the game changes the effects of certain actions, or adds new actions, depending on certain pre-conditions (event) being unlocked (or being completed in a certain way). What is interesting is that the unlocking of said available relation types is dynamic, meaning that some actions that have extreme effects, and which affect other attributes (not only of that single character), are only doable after certain conditions are met, shaping very diverse and coherent experiences based on simple checks[43].

[43] There is a problem of scope to implement all the different interactions and preconditions, which this game surpasses by being a text-based adventure, and it is undesirable for this project to limit when certain actions can be taken, putting in a dire spot the whole unlocking-actions-by-progressing mechanic, but nonetheless it was considered worth noting, as the approach to dynamism should be useful to shape our own.

4.7. Middle-Earth: Shadow of Mordor

Middle-Earth: Shadow of Mordor's nemesis system attempts to recognize player action to accept and respond to it, dynamically, with a narrative beat, something that comes very close to our objective, albeit with a much more limited violence-centric focus.

The idea is that when the player confronts an orc, depending on what they do, they get different results that will affect their future encounters. As the player progresses, the entities they have had more interactions with, the ones that are supposedly more interesting to find due to the shared storylines, are manipulated to end up closer to the player, increasing the chance to find them again (GameMaker'sToolkit, 2021).

Moreover, such value also defines how rare the most interesting interactions are to be applied to an orc. Those interactions can be things such as coming back from the dead, or escaping the player through a burst of strength, and some are made extremely distinct by limiting them once per playthrough.

Outside of player action, the system also mixes up possible combinations using an internal hierarchy, through which different orcs can challenge each other in deathly matches to go up the ranks, affecting their position and influence (although those interactions can also be a result of their performance in regards to the player).

Additionally, there is a hidden web of relationships between orcs, be it as natural interactions from opposite views (rivalries), as a predetermined relationship (like blood-brothers) or as randomly generated events (traitors that get confronted by allies, for example).

Finally, if the game detects that a player presents the same kind of behavior constantly (be it to always pardon or always kill), the game will send events especially targeted at confronting said approach, not with the objective to change it, but at least to avoid a feeling of staleness.

4.8. Minor cases of study

4.8.1. Road 96

While not much information has been found about the development of this title, the limited playtime invested in analyzing its narrative treatment has led to two interesting observations:

- Interactions with each character were disordered and procedurally generated, but regulated nonetheless by certain conditions that determined with which one the player would interact at the next beat, such as calling a taxi or having started a specific questline. Those, however, did not seem to be definitive, only raising the probability.

- In some instances, starting a certain questline would mean having a fixed set of missions every time that character was encountered in the future, until it was completed. It was worth noting, then, this pseudo branched pattern in which nothing had a predetermined order, but within the chaos, specific pieces would either be forced or more prone to appear as the player gave inputs, without ever shutting the randomness of the system.

4.8.2. Crusader Kings II

In *Crusader Kings II* (Lucat & Haar, 2015), characters alter their behavior based on a series of personality parameters inspired by the seven deadly sins and the cardinal virtues, including special interactions of disliking those who possess the counterpart defect of a virtue, and vice versa. Although this is an interesting approach to base a procedurally generated system on, it does not give much control over general personality, but the dynamic itself is worth noting.

Being based on a medieval setting, another factor is computed, which is social status through the use of family members as pawns to gain power, which could make a division between personal and social status, regulated by how much a character has interacted with the player.

As mentioned in other sections, though, each character should have their own way of dealing with the player, because canonically, they do not know who the player is, so their judgment is based on world rumors (generally accessible information) / social status.

4.8.3. The Church in the Darkness

The most relevant idea that was found in this game is how it contextualizes its decisions: the player's reasoning behind each decision taken changes in every playthrough due to how the information is presented, even when the decisions themselves are the same and have the same consequences (Rouse III, 2016).

This led us to think about how the system would benefit from being able to present situations differently, accounting for player action without changing the core narrative progression. In essence, we would require a functionality that in-takes world contexts, and then choses the appropriate way to present the narrative, while not forcing spaghetti code of conditioners inside conditioners.

4.8.4. Hades

Research on Supergiant Game's last title proved to be quite useful on the more technical aspect of the project (PeopleMakeGames, 2020). The way its dialogue is managed attempts to provide constantly unseen lines while adapting to the game context the player immediately comes from.

To achieve this, each conversation is stored into a pool, from which one will be chosen at the start of every run. The key aspect is the multi-tier priority layer system, which divides each conversation to get the most relevant one out, similar to what *Façade* does. Noticeably, it also allows for dynamic priorities, as something may become important only after a certain event, for example, having a chain of conversations about the same topic, all together if there is not any other more important to pick (not that every string of topics have to be immediately delivered one after the other). Finally, once the election is done, that conversation is discarded until the rest are used up, although the compromise could be made to leave some available in case they are appropriate to another context in the future.

4.8.5. Outer Wilds

Outer Wilds (Outer Wilds (PC Version) [Video Game], 2019) proved to be a very interesting case example, as it does not make use of procedural narrative, but is designed in such a way that the player can uncover the different challenges in any order they want, as long as they have the information (and at some times, ingenuity) to figure out how to progress.

The resemblance to how we want to dynamically construct a narrative with a tight authorial grip foresaw many meaningful conclusions, but the context of the game makes it difficult to extrapolate much: the avatar is stuck in a 22-minute time loop in which they choose a destination to explore, and act to uncover knowledge hidden within the world. This approach works because by getting killed at the end of every loop, the player's position is reset, so there

is an incentive to constantly change objectives and explore the least frequented places to see new (an easier to find) things.

If such reset did not happen, which is what we will assume since the project is being developed for more generic scopes, two options seem the most probable:

- Discoveries would be limited by re-tracking: be it due to lacking information, lacking abilities or world events, zones would have to be revisited to reach completion. This approach is meaningless to us, because it falls into the hands of level designers, not narrative designers (in essence, the distinction of needing a pre-condition to access something is already considered for the system, and beyond that, it is a matter of framing; it is not like there is no relationship between level and narrative design, it just out of place for the project).

- All information would be available from the beginning: if this was the case, the system would only need to keep track on what is uncovered and what is not, which was an already expected consideration in tracking the world state.

In conclusion, while *Outer Wilds* does an interesting dynamic treatment of its narrative, it only is a clever trick of the loop structure rather than anything applicable in a general context.

5. State of art

In the State of Art section, there will be an extensive exposure of the different explorations of narrative-related systems or design insight that became pertinent during the research process. This served as the primary point of reference when accounting for the system's structure, acquiring second-hand experience from different perspectives on the field.

Similarly to the 3. Theoretic framework, unless contextually appropriate to the section being discussed, details on practical implementation will not be discussed here, as this serves as a compilation of interesting perspectives that inform our understanding of dynamic interactive narrative systems, unrelated to our goals.

Another important factor to keep in mind is that the knowledge gathered in this section resulted from a myriad of projects that had their own unique restraints each, most of the times under professional schedules. This consideration implies that their implemented final results are not necessarily the best possible, or ones that align with this research's vision, or even that they are compatible with our own limitations, and were probably heavily influenced by their own constraints. The weight of such consideration will be later discussed in the 6. Technical discussion section.

5.1. Ken Levine

Ken Levine's talk about narrative set pieces is one of the most valuable points of reference, as it inspired the initial approach of this whole research before any documentation process was conducted (Levine, 2014).

In it, he proposes a system that focuses on player driven replayable narrative gameplay. To do so, a faction system is proposed, inside of which there are certain NPCs called "The Stars". They are distinct characters due to having assigned a set of passions, which are the world states they care about in relation to the player's actions. Those should be clearly understandable, and constantly updated based on how the world is affected by the player. Each Star would have a macro passion bar, as an average from all passions, which defines general animosity; when certain thresholds were crossed, certain behaviors or events would be triggered.

Based on the Star's macro passion's average, the Drone's, the non-distinct NPCs, would derive their behavior to allow for highly populated, coherent worlds. Other entities could also exist

outside the system, like unaligned Stars, those that do not care about faction affinity, just the actions committed, or, extrapolated to a broader context, that care about any specific manually set conditions. It was also proposed to have certain macro passions that affect the global relation of many factions with a specific one, or one that is immutably opposed to the rest of world inhabitants.

Expanding on this basis, Levine introduces the concept of hidden (yet still functional) passions, those not revealed at first, only being accessible after their macro passion bar reaches a certain point, as to simulate the development of a relationship. Finally, he proposed the creation of a pool of random passions, from which a certain amount is chosen per playthrough in order to incentivize replayability through procedural generation.

Through the examination of this system, many functionalities and considerations were noted:

- We should be able to represent interest separately from affinity, since agents should be capable of setting aside their differences in order to achieve success of some kind (even if the way humans go about it may differ depending on how distant the relationship has gotten).

- Being held in a high regard should mean that characters, or at least the most trustful ones, will accept certain actions that, in this particular system, go against their passion, and might excuse the player action's, attributing them to unknown circumstances. Incidentally, Levine mentions, in a particular example, a buff that makes positive regards gain an extra value each time, as a way to simulate social confirmation bias through group affinity.

    - This would lead to implement a subsystem that considers character personality, and which can alter passion value modifications based on it. This way, for example, we could reflect the trustful nature of some people through internally losing less points once they are in good terms with the player.

    - Inversely, if the player has a bad reputation because of group affinity, but a specific character's passions would make them positively interact with the player, it seems natural to assume they should have a negative opinion derived from that group affinity, but which should be reverted in some way once the

player "proves" their true self. It was also considered whether a boomerang effect should leave the personal macro passion at positive, since it is natural in this kind of circumstances to feel like one has "truly gotten to know" the other, when in reality it was just that a misconception was cleared.

- In the hypothetic that Levine presents about marriage, in which choosing one of the partners will imply the nullification of the other wedding candidate's benefits, we reflected on how not all of them should, as not everyone reacts so extremely to rejection, or at least not in all facets of interaction (it was proposed with the idea of interesting choices, but that should not take away from having both options; again, something that maybe should be controlled through personality traits, and then possibly determine different thresholds for each benefit / effect we may want to cap off). On top of that, it also made us think about how hurting someone's interest does not have to always mean they will hate you for that (possibly codified in altruistic or empathetic traits), and that such reaction can change over time, be it because the tension dissipates, or because their relationship with the player changes (be it from other NPC's influence, which can become an extremely complex web of interactions if we allow characters to develop their own personal dramas in a full-fledged manner; or due to player action).

- Relating to macro passion value's modification, those should be considered differently depending on both the character's personality and how aware they were about the player's situation. For example, if the player did not know them, and they act against the character's interests, being something not personal, most will react less extremely than if they had a good relationship.

5.2. Dynamic quest generator

A dynamic quest generator was proposed and tested at a small scale in 2014, one that resembled many of the superficial characteristics proposed in this project (Lima, Feijó, & Furtado, 2014).

The system was based on a hierarchical quest structure, such that they could dynamically divide and extend them as much as desired. Within each quest, there are two types of connections: set or individual key player actions (events), with each leading to another node once completed; or other quests.

Each quest had an instance of a planner and a monitor. The planner, cutting many corners on the technicalities, receives predefined actors, places, objects, paths, locations and hierarchized goals. From those, every time an action impedes the completion of the goal with highest priority, the planner will re-calculate which goal the player should follow, and based on how the quest is completed, the next connection will be determined.

The monitor checks that everything is according to the created plan, for example, that the conditions that lead to the provisional goal are reachable, or that the result of a subquest instantiated in the middle of the current one does not change the course of its said parent quest. If a discrepancy occurs, it asks for a new plan to be generated, one that fulfils the same goals, if possible, but with a different set of actions to be done.

The conclusion derived from this system is that despite taking a very different approach from ours, it gives an interesting point of reference in terms of implementation, but almost more importantly, it brings the idea that narrative lines are not confined to constant change of objectives, but that we can redefine it while just changing the context. The given example for this was that when the chance to acquire an antidote was lost, the planner would check if there was another one available, and send the player there to fulfill the prioritized goal. The thematic depth this concept can bring is undefined, but it was noted to be relevant for consideration.

5.3. PropperWryter

Described to have a structural approach to narrative (Gervás, Concepción, León, Méndez, & Delatorre, 2019), PropperWryter is a narrative generator system based on Propp's *Morphology of the Folktale* (Propp, 1968), which is divided in the following modules.

1. Plot driver generator: builds the sequence that composes the story.
2. Fabula generator: instantiates empty characters with their particular actions.
3. Casting module: assigns particular characters to the created entities.
4. Textual rendering mode: converts data into a text-readable format.

5.4. STellA

The Story Telling Algorithm is a system that simulates an entire world, and makes up a story from the interactions occurring within (Gervás, Concepción, León, Méndez, & Delatorre, 2019).

Such simulation comprises a physical, social and mental layer being kept track of; from those, multiple sequences are obtained, and from those, the narrative layer discards the ones deemed uninteresting, and sets parameters to keep developing the accepted ones. Said layer used explicit constraints, user objectives and narrative curves with the end of controlling how the more abstract elements of the narrative would be perceived to be developing, and to regulate them adequately. It is mentioned in the paper, however, that the implementation of STellA was only partially successful, as it failed to distinguish "valuable" or "novel" stories.

5.5. Charade

Charade relies on character simulation to study how character affinities can believably interact (Gervás, Concepción, León, Méndez, & Delatorre, 2019). For it, four levels of affinity were built, ordered as foe, indifferent, friend and mate. On top of that, it uses a fuzzy approach, which means that there is no constant change when oscillating between the limits of a category and the contiguous one, since said limits are diffused, not based on a hard-coded value. The system can also consider relationships non-mutual, except in the case of mates, were it is forced to be.

Said affinities change based on two mechanisms: a lack of interaction will direct the relationship towards indifference; contrarily, an interaction is proposed from a character to another. In such case, the proposing character's only available actions are those constrained in their affinity level, and if they are not accepted in the receiver's level, the interaction will not occur, and the proposer's affinity level will decrease (the one exception to this rule are foes, which act independently from the other part). If the action is valid, it runs whether it is to be accepted or rejected, increasing affinity for the former, decreasing the proposer's for the later.

5.6. StoryFire

StoryFire is a system that attempts to convert real-life events into stories, and follows the storifying computational model, starting with the identification of a narrative thread around the events, selecting a plot structure that fits to some degree with the chosen thread, mapping thread characters to plot characters, and ending with the generation of a readable version of the story (Gervás, Concepción, León, Méndez, & Delatorre, 2019).

5.7. Delatorre's suspense-driven system

Delatorre (Gervás, Concepción, León, Méndez, & Delatorre, 2019) mentions a suspense-driven system, in charge of regulating suspense. The program does so by revealing or keeping information based on the inputs of the previous scenes, outputting the next scene's results. Suspense is quantified through a number, while the input and output are blocks with information about characters, objects, environments and facts, that are analyzed based on a weighted corpus, that stores the value of suspense of each object.

5.8. Lume

Lume (Mason, Stagg, Wardrip-Fruin, & Mateas, 2019) is a system designed with the thought of creating procedurally generated narratives in a scene by scene case, and the intent to both stay consistent and allow for player agency in the development of a parametrized and partitioned plot.

Their approach is considerably different from ours, in terms of what degree of control the system has over the narrative, but we took some insightful notes of how they envisioned it:

At the most general level, each scene is composed of three types of nodes: the base one, the parameters of which depend whether the scene is selected or not; the choice nodes, which register player input; and the continuation nodes, which simply advance the narrative until the next node. While no node structure was planned to be used, this reference was kept in mind in case it could be reused or reimagined in other contexts.

Dialogue, while something we will completely ignore, is said to be procedurally adapted based on specified structures that are altered based on the events of the game, needing a character-

centric specificity that does not create a dissonance between what an NPC is supposed to know, and the tracked world events. We thought this could be paralleled in our system as storing the information of certain world conditions and states within the characters, to add a layer of believability, as well as a source of conflict.

Also very interestingly is how the world state is registered, and how entities access the records of what has happened: it is a list of events, the latest ones being appended at the end, and then checked, safely maintaining a record of all activity instead of merging it in a single undescriptive state. It was worth nothing that keeping this system raw would most likely, however, be excessive and could cause performance issues if multiple checks on the list were done at the same frame.

5.9. Twine

Twine is a node-based non-linear story editor that converts its content into HTML format (Twine Cookbook, 2021). At its most simple level, each node contains a written context, and multiple options for the player to choose from. Through the use of local and global variables, as well as the pointers between nodes, it controls the narrative flow. While accessible and versatile, it is a very simple and generic conjunction of elements that we will include in our project, but which have no arrangement that lets us stablish any useful structure as reference.[44]

5.10. Inform7

Inform7 is a text-based programming language that uses natural language syntax that compiles into a program for user interactive narrative (Inform7 Documentation, s.f.). The atomic components of the language are assertions (facts, like “X is in Y”; implicitly or explicitly in them, properties and values are defined), actions (mechanics, like “lift”) and rules (events, like “Under X circumstances, if Y happens, now X is true).[45]

[44] Similar tools like Ink, Articy Draft or Yarn Spinner have been also taken as reference, but since all those systems, like Twine, focus on providing a platform to create authorized content, it does not quite fit with the core of this project, that being the dynamic control of the narrative, not so much the predetermined branching of the story

[45] For a simpler implementation of this kind of systems, it is advised to check the Zork I repository.

The world created by Inform is populated by entities that are created through assertions, and behave accordingly to the defined rules. In both cases, when a change occurs in the world, it is not altering the object itself, but its properties, which can be shared by both the entities, and the rules. Through this compatibility between high-level and low-level actors sharing the same context, we thought of adopting some arrangements that Inform does within its system:

- As it was to be expected, default behaviors will be an absolute requirement.

- There were observed to be some basic common routines, like showing a description every time the player enters a room, which could be useful to internally manage the characters; we would have general rules governing everything, that, while not mandatory under all circumstances, would control the default progress of every character without having to configure specialized paths for common entities, despite their difference in traits.

- They way in which "kinds" are handled in Inform had been superficially observed to be as everything being of class "something", then having an enum attached. Separately, the system keeps track of a tree of all types of kinds, and checks where the enum type is when having to take a decision involving the parent classes, but with all objects being of the tree's base class. This is done in order to have large scale inheritances (the common example being thing < animal < person < man / woman), and being able to tell all the levels an object is part of, because of the way Inform handles instructions. To our system, this could be a very useful approach if we ended up making hierarchies out of multiple levels of abstraction, which considering theming, behavior, character traits… seemed a fair possibility.

- The way Inform treats light is such that for the player to be able to see something, they need to look at something visible, and then be in a lighted room, or have a source of light. It was considered relevant to note this kind of double-valve check in different aspects of the world simulation when implementing our system, possibly requiring inter-communication to function.

- Assemblies are an interesting type of object in Inform: they are the conjunction of an entity of a certain kind with another one, be it the same kind or not. This construct can

be done on the "kind" level, or at a particular individual one. It is interesting to us because it allows us to generate relation-based behaviors, overwriting a character's behavior by the one dictated by the assembly. It would have to be seen, however, whether such control would need its own manager, or if it would fall within the behavior manager / rule dictionary.

What became quite insightful of this system was its treatment of objects, as it can create a correlation between two of them, but by default, each property can only be related to one object. An example would be that an object that serves as a container can have another object inside of it, and only one, but it can also have an object on it. This would be interesting in terms of limiting character priorities, as focusing each NPC's interests allows for more specificity and believability[46].

Additionally, in the same way that *Inform7* categorizes each element it tracks, character traits or behaviors could also be codified by an enumeration variable that automatically redirected the character's behavior to a specific script (maybe using a behavior manager, which then raises the question of how many layers of behaviors or theme control would we need, and in which order they should act).

Another interesting feature it has is how it relates properties, as in "X is at Y", where a relationship of containment is stablished. From this concept we could extrapolate a map (the programming structure) of NPCs (or even locations or items) and their type of relation with the subject object, which would then combine with other factors to choose a behavior from a dictionary. The idea is that, by itself, the relation would not define the behavioral output of the character, but it would serve to maintain the result's believability. An application of said approach would be, for example, to define what kind of dynamic changes the character could go through as the story progresses (such as, this being a very reductionist example, "this character is vengeful against X faction's NPC"), as a restriction measure that defines the character without compromising the possibility of change. Incidentally, we could define properties that explicitly changed those higher-level rules, and even try to work something out to speak about the contradiction (or at least conflict) that is generated when a character acts with different standards, due to a change on their beliefs, at some point between two instances.

Another facet that the high-level rules could encompass would be combinations of rules for specific outputs (both at an NPC general and specific level, although it is unclear where that

[46] After all, our reality is one of limited resources to spend on unlimited needs or wants

limitation would lay currently[47]), as it would make sense to allow for X to elicit a positive response, for Y to do it too, but not if X and Y occur at the same time.

While looking through *Inform*'s documentation, it was also considered to implement interpolation between behavioral changes, that is, to have a transition between a character acting in two different ways. Since it would still be desirable to have in-the-moment spurts of change, a variable should be destined to define whether a behavior alteration should be reflected immediately, if it should wait until the next time the player and the NPC are separated, or if it should have a timer attached to it.

5.11. StoryNexus

StoryNexus is a narrative qualities-based world and narrative builder (FailbetterGames, s.f.). The building block of this system is that all of its characters are defined by a set of qualities (be it numerical or numerical but substituted by a text), which can go from an identifier of a story thread to how many types of a specific object are possessed. Among them, they are divided between statuses and things. Additionally, quality numbering can be additive or pyramidal: respectively, increments of one; or going from the first unit to the next for a cost of one, but having a cost of two from the second unit to the third, three from third to fourth…, until the quality cap is reached. Qualities can also be set directly into a certain value, which can be a number or another quality's level. Managing of qualities is decided based on their type.

The system also has a quality manipulation so that if an item has an effect of +1 to X, and X is 0, when equipping said item total raises to 2: +1 from having the quality exist (as 1 is the minimum value of any quality), then +1 from the object's effect. The justification for this was to avoid abuse of passive effects, since gaining animosity from a certain faction should not be corrected by simply removing the item, as the characters should have the "memory" to remember the player presented itself as someone opposed to their faction. This approach was deemed reasonable and practical, and thus, taken notice of.

[47] Going out on a bit of a tangent, we could probably do a general template that defines the "standard" NPC stats, which would then define behavior, and make a child class that overwrites the values based on that character's specific personality

Another interesting aspect of StoryNexus is that it allows area-restricted equipment, which for our case can be extrapolated to actions, too, adding another layer to the behavior of NPCs that was not previously considered.

5.12. Encounter manager

Daniel Kline proposed an algorithm for a dynamic narrative system called the Encounter Manager, which, inspired by pen-and-paper Game Masters, presented story threads dressed up as sequential encounters.

Story threads are made up of story knots (the individual encounters), and each knot can be divided in two types: those that progress the narrative, and those that serve as a breather or set-up for future plot advancement. All knots contain a variable that codifies the type of scene (while it is referred in the paper as the theme, it differs from our usage of the world, here being used as the tone / feel), and its pace, as an integer from 1-10.

When a story thread is completed, the pace drops a pre-designed amount (to release tension), before going to the next story thread. Progression is decided by both player input and narrative context. In some cases, certain beats could be forced at particular spots.

It is also mentioned how having a final goal in the narrative helped in constructing the breathers or minor moments (which do not imply not having action) towards that ending, such as having to escape from something bringing the player towards their goal, as a transition instead of just "the characters walked there".

Because the game's purpose is the generation of a story, it was very useful to test it by leaving the game running, generating content, and then analyzing it manually to identify patterns that should not be generated by design. While our system does not have this approach, the idea could be adapted to facilitate error detection.

They remark in their conclusions that drama managers are not made to allow player freedom, but to have pacing quality that does not require an authoring burden, because they limit what players can do based on what they have done. Since we want to enable as much player expression as possible, we ought to stablish (or accommodate) a less restrictive system that allows a change of attitude if the player so desires, instead of restricting them to follow an Aristotelian (for example) narrative without changes of heart that alter the structure. In essence,

the idea is that we could avoid getting stuck in a continuously scaling conflict if, by acting, we manage to deflate the conflict preemptively, thus altering the pace in favor of thematic coherence.

This, however, could end up causing a diegetic conflict if the system cannot justify a player's recurrent change of paradigm, and nonetheless overlaps with other character's accepted behavior spectrum[48]. What is more, if by trying to open the overall narrative pace, allowing more fluid storytelling, we end up with constant resets of the status quo to a state before conflict escalation, we run the risk of entering an infinite loop of inaction. Since the idea is that the system dynamically controls character behavior based on world and character state, unless we define an absolute truth (which could serve as the overall theme of the artwork, even if it is disingenuous to think there is such thing for most aspects of life), it could be the case in which the system constantly loops between a sequence of behaviors, never scaling the narrative[49]. While this could be prevented by fixed change factors that force conflict scaling, deeper thought must be given to the relation between agents and drama manager to avoid this kind of problematics.

Finally, while it was not mentioned directly, this system did spark the idea about setting up storylines and having them activate by player action, be it guiding them there with audio queues, visual indicators… In summary, there would be a distinction between "active plot" and "standby plot". The downsides of this approach would be the unnecessary load of generated content that might not be explored, and the workload that it would suppose to implement both the signaling systems, and the controlling elements, such as timers and checks to manage each plot. However, this content could be reused to generate player-external narrative that provided a living world beyond player action, or even to create a mythos from which to base the world.

[48] Basically, not all character should be able to change their character completely: we want a dynamic system, but we also have to stick to a minimum of believability when making changes, especially in complicated subjects like human character.

[49] The idea is that we part from character state X to Y, Y to Z, and eventually, W to X. An abstract example would be "character A blames society for their failures" into "character A realizes their laziness limited their success" into "character A realizes they are trying to access a despotic system" into "character A fails to achieve their goals" into "character A blames society for their failures". Because the truth of whether character A could have taken action in such a way that they would have accomplished success, how responsible the social system is for enabling their failure, and the level at which this circumstances could have been controlled or predicted, does not exist, or at least, is out of human understanding, it is unlikely we can code a system that makes sense of this kind of situations.

5.13. DINAH

DINAH is the / a Dynamic, Interactive, Narrative Authoring Heuristic to construct textual stories from a pool of short stories with preconditions that enact their existence, and postconditions that other narrative pieces are spliced with (Ventura & Brogan, 2003).

The moment-to-moment state of the story is stored in a set of vectors, and iterated continuously to revise the condition-based system that controls the construction of the narrative. Each story clip is defined with an ID and text that defines possible actions and reactions, as well as state vectors for the conditions. Once a story is completed, the story clip database is ordered by relevance, and the next fragment is chosen; if many coincide in priority, the user gets to choose from the four more relevant ones; instead of being presented as a raw choice, however, it is masked by a story event where the player discusses their opinion diegetically.

The internal logic that prioritizes story clips follows Branigan's model, briefly summarized as the following linear structure:

1. Introduction of the setting and its characters
2. Introduction to the character's context
3. Inciting incident
4. Protagonist response to the inciting incident
5. Emotional tribulations
6. Conflict outcome
7. Reactions to the outcome

Depending on the step of the model currently in, the selective algorithm will prioritize story clips differently.

Among the system's matching conditions exist character states, inherent to each character, which indicate mood or relation to someone or something else (including relations like having visited certain places).

Finally, the system also manages temporal coherence, defining time through four dimensions that also are considered to control narrative progression:

- Major Sequential Time: separation between phases of life (before becoming a master, one has to be an apprentice).

- Delta Time: limitation that ensure only one line of action can occur concurrently (one can be the apprentice of one craftsman, but not of many, while being able to act in other aspects of life).
- Mutually Exclusive Time: manager that culls story clips that can no longer be realized when reaching out for a new story clip in the database (one cannot meet up with a friend if they left to study abroad). This manager acts during the play of a story clip, not once it has finished.
- Minor Sequential Time: organizer of a sequences' events at an action-to-action level (one cannot make a sword without first gathering the materials, getting the required tools to work on it, melting the metal, cooling it…).

As each of those dimensions advance, the pool of story clips is reordered to separate usable from unusable ones, back and forth, as it is possible that something that could not happen in the past become available in the future.

It is mentioned in the conclusion that it is important to not overload the narrative with constantly occurring events, something already considered, but reemphasized here, both to avoid overloading the player with information, and to pace the story for better engagement.

5.14. WAWLT

WAWLT (Why Are We Like This) is an emergent narrative-centered project developed by Max Kreminski (Kreminski, 2020). Its focus lays in having two players make a story about a mystery case, with each character having a set of story shifting patterns that allows them to make a judgment, creating a perceived reality for each of them that will eventually conflict with the other's.

The process of generating the story begins with a scenario being generated, with a collection of characters, institutions they can be part of, and relations between both characters and institutions. From there on, a scene loop repeats until the resolution: players choose their goals from a list of predetermined actions, and which characters they want to have participating in the scene. The system provides suggested actions, according to the established goals, and they culminate in an effect that changes the web of relationships between characters. The acting process is repeated until the players decide to change the goal, at which point they will close the scene to go to the next. As each scene occurs, the characters that were left out also act

autonomously, partially guided by the established player goals, but with a hint of randomness to generate more independent behaviors.

The control of the world state is managed through a save file, with a list of its containing entities, that stores the following information:

- Characters: name, role (equivalent to their job), two values and two curses (essentially, simplified personality traits).
- Relationships: impressions and role-relationships of each character, expressed as a numeric value of overall attitude (plus and minus).
- Impressions: list of the events that caused a relationship defining value, which is the values used to calculate the net relationship result, with a maximum of three positive and three negative values per relationship, the highest values of each being prioritized when new values get generated. Impressions also safe the value registered in the event.
- Projects: property that defines an on-going character effort (which can be shared by more than one agent), mainly composed by its contributors, the project name and a numerical drama level.
- Institutions: represents the possible affiliations of a character, but is mostly used for backstory generation.
- Events: the recordings of a character's action, kept in record to simulate character memory. They are composed by a type, the performing actor, a description and a timestamp, as well as an optative list of tags.

An interesting aspect of the system is the character backstory generation, which is constructed completely apart from the systems that take place in-game, instead, using a series of concatenated non-available game actions – ones that usually take over long periods of time (like publishing a book or graduating) – that build a "shared intellectual and social history of the community for several years prior to meeting", as the author points out.

About the in-game available actions, they are defined by their preconditions and effects. Actions are divided between external or internal, which define whether their effects should involve elements of the world, or whether they are directed to themselves (their internal states). For this second type, it is designed to trigger a process of personal change over time, until the character reaches an affinity alteration, but if the action of introspection is registered too long after the inciting events took place, the introspection will not occur. This is prevented with a

special introspection action, but was implemented to avoid inconsistent behavior of characters thinking about past events long after they had taken place.

5.15. StoryAssembler

StoryAssembler is a narrative system that takes a story fragment from a so called fragment library and delivers it to the story spec, the equivalent of a scene with a precondition to be followed with, to create a choice-based narrative (Garbe, Kreminski, Samuel, Wardrip-Fruin, & Mateas, 2019).

After an introductory scene is selected, the system is responsible to fulfill the story spec's goals by choosing as many fragments as necessary to push the scene's values towards said goal values, entirely calculating a best path for the player experience. For this concatenation to take place, the system gives each fragment a score, considering how much the continuations of that fragment helps to reach the goal (this is calculated through the variables that they would modify in the blackboard). This transition can be accomplished both through body text or player choices. The fragment path to reaching the goal is only recalculated after a player input, to improve system fluidity. Scenes end when all specs have been fulfilled.

Some specs can be marked as persistent: they will always take priority, but will never be considered satisfied until the scene ends, which is used for repetitive actions and flow control in specific types of scenes.

Finally, fragments can become unavailable or available depending on certain conditions, which range from time, to performance, to context (some can be chosen at any time, others only after certain labelled fragment/s have occurred, others under both conditions).

5.16. Jason Merrin

Jason Merrin implemented a dynamic interactive system, similarly limited to our conditions (Merrin, 2013). In his project, the goals were to accomplish AI that behave meaningfully with autonomy; a storytelling AI that looked for conflict, and stopped that story when the conflict resolved; finally, supplying a basic game that showcased those features.

The author cites a paper about narrative roles, and explains how to parametrize them; while the idea of stablishing classical roles, as useful as it could be, did not mesh with our approach, it was noted the utility of defining the reactions of each character based on patterns displayed in the story's plot. If anything, it could serve as a powerful affordance in case that making all reactions manually explicit becomes too cumbersome for the development of the research.

It is also mentioned that besides the affinity mappings, a "current emotionality" value could be implemented to dictate how unstable the character is at the moment, which could alter the chances of exhibiting certain behaviors. While this brings more interest to interaction, implementing a dimension of time-based individuality, it also could end up being a whole system of its own, as we would have to control what and how the environment affects each character's mood, and how to display it, which could easily go out of scope.

Finally, the system in charge of controlling the story was made so that it would develop with the use of actions, ranked by their emotional weight from 0 to 5; the higher the value, the more likely to be recognized as relevant story beats, concluding the story once the character assigned as the protagonist had gone through three important actions.

Through this simplified method of story interaction, an accessible framework was provided to add new actions: using a .txt file, where each line being basic data, with a predefined length that formed a cluster that represented the action. The code imported the data from said file, converted it into internal structure, and used it to affect the primal values that defined importance and outcome of the newly defined actions.

5.17. Character-centric narrative proposal

A classic approach to managing character relationships, one that does not exclude other systems, consists of integer parametrization of different aspects of the agent. This is the most extended approach due to its facility of use, read and implementation, and usually comprises a range from –X to X, with the correspondent thresholds to define ultimately the various character states. The case has been made to take that character affinity and measure it (although there is no reason to think it, again, excludes other parametrization techniques) to define narrative structure (Zhu, Ingraham, & Moshell, 2011)[50].

[50] There can be, after all, a strong relationship between mechanics (actions) and narrative (Dubbelman, 2016).

Basing their hypothesis in interactive performance projects, the authors propose control of the narrative based on the subversion of the status between two characters, and the power struggles between them. This would entail keeping a record of character confrontation (possibly dynamically controlled based on player action), and forcing situations in which the involved characters changed their power dynamics based on a paced schema. It is proposed to understand actions as forces, being resisted, blocked, pushed through… not concluding a final value to add or subtract from a bidimensionally ranged array, but as a series of interactions to subvert.

While this approach certainly generates character drama, it should probably not be left to carry the entirety of the narrative structuring, as repetition would result in predictable plots. Additionally, by depending always on the same kind of interaction we would be cutting off the possibility of broader thematic discussion, as power dynamics as a relation-defining trait would cause thematic noise.

Whichever the case, the possibility of having multiple sub-systems (within the narrative structuring department) would assure a variety of very specified and relatively safe structures to be presented. This would, in turn, imply the need of a sub-system manager that took into consideration the entirety of possible structures, the current plot and player state, and the capacity to decide which structure to present next[51].

5.18. Richard Rouse III

Richard Rouse III makes a strong case for a different kind of management system in one of his GDC presentations, on how reacting to player actions does not have to equal setting a group of values in multiple axes, and have the inputs move their value towards one of the extremes, but instead, having each action progress the story purely based on its own effects[52] (Rouse III, 2016).

To our system, an implementation that followed this idea would be to separate between parametrized events and predefined events, so that certain paradigms of the entire world, or any lesser level, can be altered after certain events (for example, if the Armageddon comes, and people have less access to primary resources, their affinity, or even type of reactions with

[51] A simplified approach could be take to simply define the individual structures within the system instead of implementing entire sub-systems for each desired structure.
[52] The example he gave of this implementation was Ken Hudson's *The Novelist*.

outside factions, will change for the worse). This way we maintain both the micro-level detail influenced by the world, with access to changing macro-level states.

5.19. FAtiMA-based system proposition

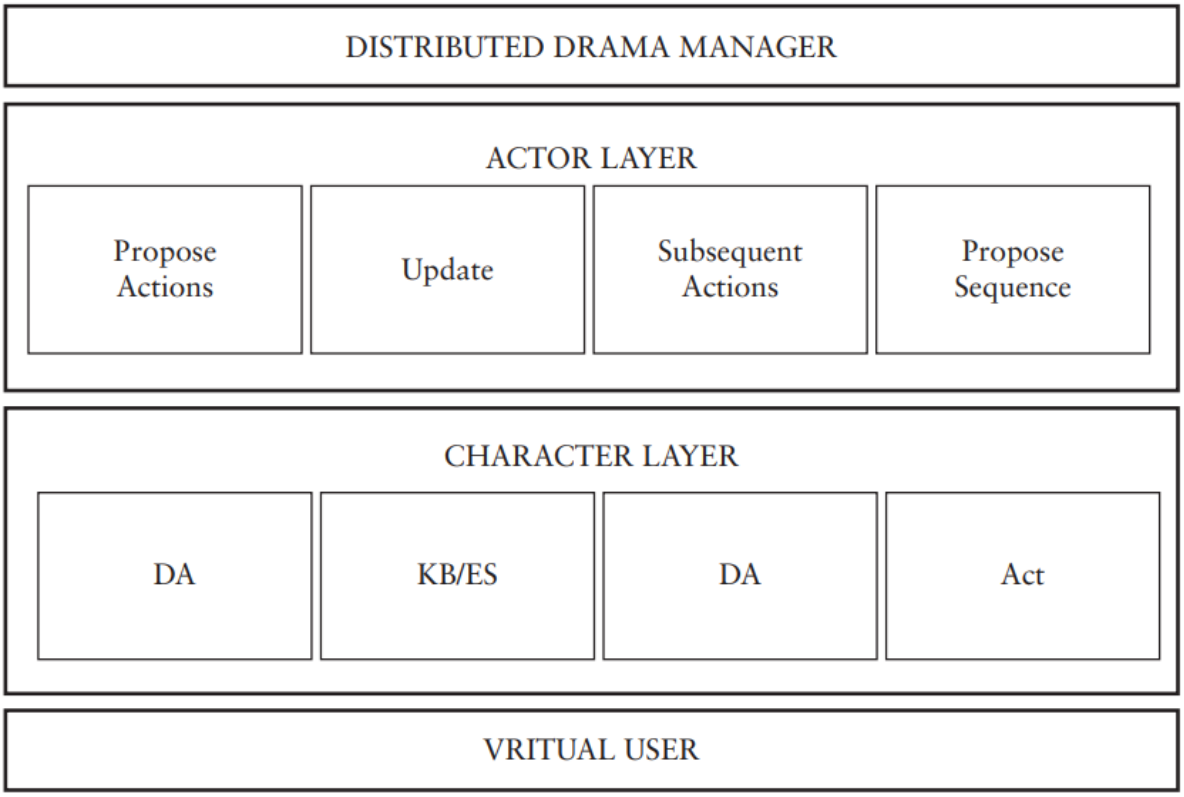
Other more restrictive, specific approaches were also proposed (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017), like an emergent narrative generator based on the FAtiMA agent architecture, which emphasizes characters building the story and informing the rest of the systems on how to continue. In it, the characters are aware of the consequences of their actions, and use that information to take decisions, but they do not know how those consequences will inform the narrative. Because of the focus it puts on characters, this approach was recognized to be ill-advised to plot-heavy games, instead driving the plot through relatively unpredictable emotional entities, more similarly to what is to be expected of emergent narratives.

This system is relevant to this research in so that it proposes characters not as copies of people, but as entities with narratively aesethetic, thematically illustrative and mimetic (of humans) properties. In it, each entity is aware of how it affects others, but not how it conditions narrative. Acknowledging that it was worth, however, maintaining focus and control over the narrative, a series of sub-systems were introduced.

- Distributed Drama Manager (DDM): system that controls both agent's intensity reaction based on narrative pace, and the type of actions that the character can do based on prior experience.
- Relevant Context Modelling (RCM): controller that balances conflict between contextually appropriate action and strong emotionally-driven action; its main problem being the definition and measurement of context to make generate an output. The proposed solution is to simplify and consider only the emotionally relevant data, which can come from events, actions and staging. It also controls how the character personality changes by the environment without breaking its defining aspects, by adapting decisions to the context.

Figure 11 Distributed Drama Manager schema

made by the authors (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017)



Later on in their book, the authors also gave some advice on how to develop dynamic narrative systems, stating the usefulness of abstracting to generalize reactions at the essential level, not the superficial one (something that overlaps very strongly with this research's thematic approach); conditional thinking, generating content based on actions to form sequences of clearly comprehensible plot events; and considering the user actions first, which was already covered by using mechanics as narrative building blocks.

5.20. HTN Planning Systems

Among the many approaches discussed in the state of art, there is one the relevance of which is worth noting at a theoretical level: hierarchical task network (HTN) planning systems. As explained in Professor Nau's lectures (Nau, Hierarchical Task Network Planning, 2016) & (Nau, HTN Planning, 2016), HTN planning systems are a way to build a tree of behaviors that are selected based on a series of conditions. The approach they take to do so is in dividing the process of reaching the goal into multiple predefined tasks, defining each major path as independent and relevant to a set of circumstances, providing action freedom and specificity to achieve very differentiated author-controlled outcomes. This has been proven to be very

effective ways to dynamically control character behavior by letting them adapt to the situation, instead of coding intertwined state machines (FEAR, 2005) & (Killzone 2, 2009).

The system begins with the definition of an initial state (in our case, possibly a combination of world state, faction state and local character state) and a task to perform (in practice, there would be multiple tasks to be performed at the same time, competing to be done through some kind of prioritization queue).

Tasks can be classified in two, based on what follows them: primitive tasks execute an action, which is a node at the end of the tree and of the particular task it is called from; meanwhile, compound tasks are followed by methods, a tree node that leads into another task to repeat the process recursively until an action is reached. Alternatively, all actions could yield failure, in which case there is no action to perform, or the system falls back to the default action (which would be preferable when aiming for defining character behaviors).

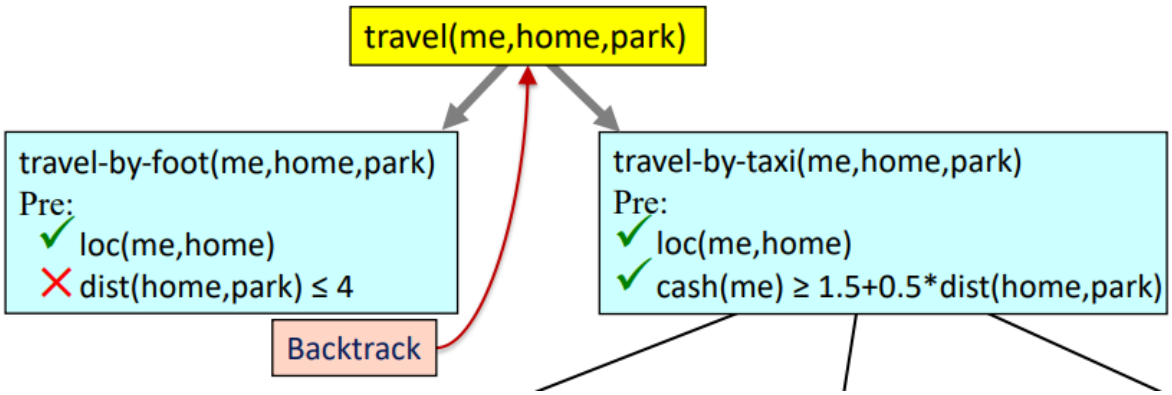
Each task receives a list of arguments that will be passed down to do the corresponding actions. Each action or method contains local preconditions and effects. If the preconditions are not fulfilled by the sent arguments, the system backtracks, returning to the task at hand, and checking whether there are any other child nodes available (in essence, other different methods or actions that could be possible to take). If the preconditions return true, the effects are executed, whether they are iterating the next step, or acting; for this reason, defining a precondition (or a set of them) has to ensure that what follows is executable, otherwise a contingency plan has to be adapted to backtrack and continue the planning search.

Every time an action is performed, the initial state adapts to register how it would be affected, so that any subsequent actions have an updated state to work with. This implies that the planning system can be executed at real time (when an action is chosen, it is done, and only when it is completed, the system will give the following step), or preferably, that it calculates an entire path under a set of presupposed circumstances, and adapts only if the relevant variables change. As per usual, such consideration will depend on the context in which the system is applied.

In either case, once a plan has been found successful, the chain of actions (or the single chosen one) are stored to be followed at proper pace. The case could be, however, in which there are multiple possible solutions. Usually, once the plan is formed (based on how the HTN tree is traversed), the iterations end, but it can be done so that all plans are stored, and then judged to pick the most effective one.

Figure 12 Example of a working HTN planning system

made by Professor Nau (Nau, Hierarchical Task Network Planning, 2016)



As useful as this approach can be, however, it is forcing authors to generate manual content to define character behavior, which can hypothetically become a development burden, which was one of the problems this system aimed at correcting[53]. On the other hand, because we are supposedly working with a limited set of in-game mechanics public to all agents, the amount of possible actions to be taken should be limited enough to not result in such situation.

5.21. Conclusions (state of the art)

As previously commented, the effects of the state of the art on our research will be explored later on, but some observations are interesting to be made at this point. From all the practical systems studied, there is a clear distinction into two main groups: pure structural systems and simulations (with narrative structure control or not). Their main differentiation is where the authoring burden lays, each with different degrees depending on their complexity and usability.

Our approach will be directed towards simulated environments, since it offers more freedom to player choice without increasing the authorial burden as badly as manual content does (or at least, it was hoped that the resulting system facilitated the creation of content partially).

In a similar vein to what the community commented a few years ago (Gervás, Concepción, León, Méndez, & Delatorre, 2019) and (Mateas & Stern, 2003), we, too, believe that the best path to dynamically controlled narrative generation is through a mix of sub-systems that, unitedly, put together all necessary elements for a satisfying result. In that sense, the vast

[53] The reluctance from asserting it practically coming from no experience applying the system.

majority of approaches taken at specific narrative disciplines, having been developed (or researched) at more or less depth, end up providing useful feedback, because they would at least focus in one of the many systems we will try to implement.

A major downside to approaching design as a Frankenstein of other creations would be that their considerations might end up limited or shortsighted for our more ambitious plans, or that they might even create dissonances between them, because their design did not account for interaction outside the sub-system itself. Managing this problem is as "simple" as designing smartly, understanding each piece's function, and putting enough time and attention, figuring out how all the parts should work together.

There is, however, a level of uncertainty generated by the lack of thematic focus on both narrative-generation takes. With no precedent to turn to, it is most wise to infer why the referenced systems were designed the way they were, and try to apply that same perspective when implementing theme-controlling structures, as well as using our prior programming experience to avoid as many fatal pitfalls as possible.

Due to all the mentioned observations, and accounting for the time restrictions, it is expected that our system manages much more diversity of information, but does so with less complexity, or less accurately. This pool of content that interacts with each other will also increase the time dedicated to testing, which points us out towards thinking about an iterative process that delivers small pieces of content instead of entire systems at once, which may leave some functionality incompleteness across all systems, but with balanced output quality.

6. Technical discussion

The Technical Discussion will be a segment dedicated to exploring different alternatives to the system's implementation, both derived from the 4. Cases of study and our own insights. It will be dedicated to discussing various possibilities and reaching theoretical conclusions, while the final implementations are left to be expanded in the 7. Technical demo section.

6.1. Limitations and guidelines

A necessary step before going into the specifics is to define what is the system's scope in terms of control, as it became obvious early on in the research that hard limits would have to be set between what the system should be responsible to do, and what would be left on the developer's hands.

Having the objective of being malleable enough that any story, under any setting and mechanics, could fit into it, strictly following the designer's intent, the system's considered features could be divided in three:

- Developer features: those are considered outside of the system's scope, and thus will be ignored, even if they should be able to take data from within our system for personal use. An example would be the saving system, or an inventory system; it is not to say there could not be thematic potential in those features, it is just not a major factor in narrative development, and will probably fall into the specifics of each artwork rather than a functionality in a generalist (in public) specific (in use) system.

- Structural features: those are the features we cannot control, but which we should be able to manage to achieve adequate performance, providing the necessary structures in which to splice the developer-made content. This would be the case for controlling mechanics, for example: we cannot predict which will be implemented in each game, but there should be a generalist framework that allows satisfactory monitoring of their effects and implications to take them into consideration in the narrative dynamics.

- Defining features: the ones relative to narrative control, the main focus of the system. Since it is thought to be used in narrative-heavy games, it only makes sense that the

narrative aspects of the game take most importance, and as such, can take the necessary liberties to work as best as possible.

Before providing a list of which features were to be implemented, however, there is, among others, a crucial principle that applies to all sub-systems, and which needs a bit of explaining: flexibility. The reason why we tried to allow as much freedom as possible within the system was, ultimately, to enable artistic expression that lead to an honest communication between developers and player. While this principle was clear, it was also mentioned that the system should empower the design vision, not the other way around, because what was fundamental was not the act of communicating, but the value of the information transmitted.

As a consequence, we parted from the idea that every system should englobe as much meaningful agent diversity as possible, and then we would build as many restrictions (if possible, independent from each other) as developers might need until a point of over-specificity was reached[54]. Through this flexibility of player agency regulation, developers would be able to find the right tuning for any specific context; the procedures to get there, however, require further explanation.

6.1.1. Mechanic interpretation

Limiting the system to a set of predefined mechanics was one of the least desired concessions to do, instead striving to allow any set of mechanics to be implemented. The problem this entails is how the system recognizes a mechanic's existence, and more importantly, how does it thematically interpret it. Neither the code structure that executes the mechanic, nor the visual pipeline, nor the button mapping, nor the context in which it is used is intrinsically defining of a mechanic in such a way that we could make a system that analyzed those to "learn" what the implications were.

An option would be to mark functions as mechanic-containing code (with a flag, for example), then compile them all to form a table of mechanics in run-time. This would be completely non-descriptive, and would not give the system any guide on how to use that information, it would merely enumerate it, making it a useless approach.

[54] At least at a theoretical level; in practice, the most limiting factor was expected to end up being time.

The closest thing we could think of to achieve dynamic mechanic-theme pairing would be to analyze the consequences of the mechanic as it is used meaningfully on the world, but for that, there would have to be a previous author work on defining each entity's thematic possibilities, and then, discern how a specific interaction is considered within that range.

Seeing how this approach is both redundant to the desired dynamism, and prone to generating poor code, a better compromise for themed mechanic implementation would be to do it the other way around: generate a thematic space for the mechanic, and then consult it to decide how the actions are considered within said space, depending on how they are used in the world.

6.1.2. Theme representation

As it has been touched upon, the vastness of theming intent has the potential to reach every aspect of artistic creation, and as a result, it is a hefty task to consider in all aspects of a hypothetical, unknown artwork.

This implementation problem is accentuated by the project's willingness to support any kind of theming, presented in as many ways and as interconnected as possible. As a result of this universality, themes ought to be represented through generic structures, rather than multiple sets of specific ones, to englobe any option the developers might want to design, complemented by multiple translators of that core to each sub-system, which in turn should also contain generic structures that embrace any kind of theme.

Dynamic theme detection was also considered, but similar problems to those of dynamic mechanic detection arose, most importantly, in how a theme is identified. With mechanics, one could say that the action itself is a certain instruction, or a certain set of effects on the world. A theme, however, is what is implied by the action, something out current technology would never be able to understand without prior definition. Moreover, when discussing theming we are englobing every component of the narrative, not just their relation with the mechanics, complicating the matter even further, and going completely out of scope.

The limits of thematic representation, then, are non-existent in type, but are restricted by manually author-defined intent.

6.1.3. Player freedom

In terms of player freedom, the system's commitment is absolute: any mechanic should be usable under any context, unless the developers actively limit it under certain circumstances.

This assertion implies the possibility that a player acts contrarily to the narrative structure, for example, killing a character that was the focus of an entire self-contained plot line, dismembering any possibility of progression. Such a result is expected and accepted, as the system's priority lays in setting the scene for the player to introspect through action, instead of trying to present a fixed story.

An argument could be made that allowing total player freedom in meaningful settings is not the same as enabling meaningful settings for the player to take meaningful decisions. While we agree with this idea, and while the system is already devised to set the player in spaces that can not only be meaningful in choices, but that encourage the player to interact meaningfully, it is ultimately not in our hands to decide what kind of interactions are meaningful and which are not. That is the reason why limitations are available to developers, but not forced upon the system.

6.2. Features' introduction

With a proper typology fleshed out by a central guideline that will inform the entire code, we can proceed to expose the features that would ideally be implemented in this project, starting with a list and a brief explanation of each, then showing their importance and implementation priority. Those have been gathered mostly from the research process, and as such, proper references will be given when discussed in depth.

- Character control: manager of character behavior when an interaction occurs.
    - Knowledge management: manager of the NPC's information.
    - Social webs: manager of multiple agent's and faction's relationships across the world.
- Data saving – loading: structures that will store the content of the dynamic narrative system.

- Debugging: structures that will facilitate the development and guarantee the functionality of the system, especially at high levels of interaction.

- Manager manager: system that ties up all other modules, for centralized organization.

    - Event manager: centralized manager of all interactions, and the sender of the correspondent responses to each module.

    - Information communication: channel that extrapolates system data into accessible content for external sources.

- Narrative structure: composer of the story.

    - Narrative memory: storage of all past events to account for them in future decisions.

    - Pattern detection: identifier of patterns, most importantly, of player action, to inform other systems of them.

    - Plot detection: identifier of emergent narratives.

    - Quest delegation: subsystem in charge of assigning a player objective to an agent.

- World simulator: system that iterates the world's autonomous behavior.

    - Mechanic control: tracker of actions being taken in the world, be it by player or in-game agent.

    - Object tracking: map of each relevant object in the world.

- Theme control: director of theme coherence and progression.

- Time control: checker of time coherence.

These features were then ordered by how much they contribute to fulfilling the project's goals:

Figure 13 Feature priority

High priority	Medium priority	Low priority
Character control	Data saving - loading	Information communication
Debugging	Information reveal	Object tracking
Event manager	Knowledge management	Quest delegation
Manager manager	Pattern identification	
Mechanic control	Social webs	
Narrative memory	Time control	
Narrative structure		
Plot detection		
Simulation		
Theme control		
World simulator		

As it was mentioned in the 2.1. Project definition segment, the system is envisioned as a development tool, and as a consequence, any visual feedback is but support to ensure it works adequately; the time necessary to gather and implement all the assets necessary for a system as extensive as ours would end up being, in fact, a detriment to its development. This is not to say we would not implement clear visual feedback, as it is necessary to get readable debug information, but it would be kept to a minimum, and mostly rely on text to communicate key information.

As a closing statement, we repeat the mantra that code should be clean, portable, usable and readable, even if we intend to provide subsystems that facilitate the extraction of information for external use.

6.3. Managing narrative dynamism

A fundamental step of the prototype's design was the adequate distinction of each system and how they fit in the pipeline. The first steps, and the base on that matter, were inspired by *The Long Path to Narrative Generation* publication (Gervás, Concepción, León, Méndez, & Delatorre, 2019), because despite the many proposals that had been made to establish how to

properly handle dynamic narrative, most of the times there was a very specific focus on what had to be controlled.

While it is true that we, too, focalize in a specific matter (themes), its nature encompasses much more from the narrative, in turn making it more of a narratively holistic framework. Following this mentality, the authors make the assumption that a better generated narrative would be accomplished by multiple specialized systems, each performing their sub-tasks while coordinately acting with each other, and propose the following modules:

- Narrative structure: subsystem that controls the order of events in a plotted sequence to form the best holistic experience possible.

- Simulation: subsystem responsible of taking the world state and advancing it through external inputs (be it from other subsystems or from player activity).

- Character affinity: subsystem that takes character interaction (or lack thereof) to define relationships between different agents, triggering different behaviors based on the results.

- Plot detection: subsystem that can take an input (be it simulated, given or generated[55]) and identify the components necessary to categorize it as a story, so that it can inform the rest of subsystems of how to reaffirm the emergent narrative that may or may not have sprouted in the player's mind.

- Information reveal: or focalization (Gervás, 2009), is a subsystem that decides which parts of the narrative are revealed to the player, in order to keep a desirable ambience of suspense and tension. It should be in charge, too, of when the reveals occur.

While not all of those subsystems were later considered a priority, they are a good place to start visualizing the key features, since most of the narrative action will center around those; moreover, because the paper was published as a recollection of how narrative generation had been attempted prior to 2019, it also provides a safe and solid ground, backed up by past, empiric experience.

[55] Consequences of stepping the world, given by the player, or pre-defined in the code.

The aforementioned system intercommunication was so important that a short typology was needed to break down the problem and avoid as many dead ends during the design process as possible:

- Secluded systems: they work by themselves, or can be adapted regardless of the system's approach. This category includes information communication, time controller and the save and loading functionality.

- Major narrative systems: in charge of building the narrative. Simulation, character…

- Connector systems: define how information is delivered and stored, affecting major narrative systems without having a major part to play in narrative control. Includes debugging, event manager and world observer.

With the different types of modular relationships in mind, and with the base from which we began designing the system explained, we will proceed to expose each feature in an orderly manner, from the most relevant ones to deciding how the system had to be implemented, to the least ones, trying to present them in a digestible but also insightful manner.

6.4. Event manager

Sustaining a living world is a complex task with many different agents acting at the same time: character behavior, player action, narrative development, world events, etc. With every system in charge of already complex processes that are intended to be as self-sufficient as possible, creating dependencies to solve the need for intercommunication would have only sprung the problems that were trying to be avoided in the first place.

The event manager is a system that is in charge of enabling communications between systems to keep module dependencies to the minimum, being our first and most important action taken to achieve effective compositional representation strategies in-code (Ryan, Mateas, & Wardrip-Fruin, 2015). It receives the notification that an event has occurred, and fires it to the correspondent listeners at the right moment, depending on whether the signal has to be sent immediately, or there is a necessary trigger for it.

The base that would be used for this system is one we had already used to develop two different games, which supported its effectiveness from firsthand experience, albeit some modifications would be needed to adapt to the specificities of this project.

Originally, each module registered under the events it wanted to be notified by, but it was decided that all systems would get noticed by all events. Despite how inefficient this implementation might seem, it frees developers from having to register each event for each system that needs it, and most importantly, it does not require them to pay attention on which are already registered, and which are not. The process of sending the information is the same, the only difference now is that, when an "undesired" event signal is received, the switch that would check which event it was would not have any defined case, would go to the default clause, and quit the function without doing anything. If in the future there was a need for the developer to add that event to the module, it would be as simple as to add the case in the switch.

On top of the signal itself, there was a need to send variables that prevented module dependency. To achieve so, it was decided to use two union variables: one for simple variables, and one for complex variables (structs that englobed combinations of variables), each with an enum that indicated the type of variable to receive.

6.5. World simulator

Following the ethos of system independence, the definition and control of the world became the first priority. It is, after all, the frame in which the story develops, thus, in which all major systems operate and cross-reference a good chunk of their information.

Ideally, it would be as simple as to make a system that keeps track of the world, and from it, inform the rest of modules through the event manager. We say "ideally", because games do not truly have a world: there are a series of concatenated areas, or levels, that are progressively loaded for the player to move around them, but they are not there until necessary, the same being true for anything contained in them.

Basing the diegetic representation on scenes, instead of a world, has been the traditional (and most resource-efficient) approach to world implementation, and it is how we should expect developers to plan their projects. In both trying to respect that, and to account for grand-level context in narrative construction, as well as designing a developer-friendly code, it became necessary to take the control of the world into our own hands.

Following the example of world generators like *Dwarf Fortress* (Dwarf Fortress Wiki, 2022) or *Inform*, we would define as much content from the beginning as possible. This is not to say that new elements could not be dynamically introduced, but the more expanded the foundation, the faster the system will generate uniquely user-catered stories.

To understand the world being tracked regardless of the currently loaded level, we must understand how to keep track of individual scenes. The first notion that became clear is that the world tracker is a structural feature: we set it up for the user, but they are ultimately the ones responsible to make it work.

The reason for this is that we cannot have the world simulator englobe the functionality of a scene manager, since most of it have nothing to do with an external narrative control tool. Additionally, if a set structure was to be established for how objects had to be placed on the scene's hierarchy, it would be a considerable restriction to developer freedom.

In order to keep the world under control, without taking responsibilities from the scene manager, then, we decided to make a narrative-oriented model that kept the most important, indispensable information of the entire world within the world simulator. As a consequence of not holding the "real world" within this model, however, there had to be a channel that enabled bidirectional communication between the game world and the narrative tool's one. Whenever one of them changed, the other would be informed and altered accordingly.

Since the model should be representing the entire world, it would most likely end up storing a lot of information, so dividing it became a given to improve code readability. It would be the most natural, then, to divide the world in "simulated scenes", and then have the game scene send its ID, to check which part of the world has to be managed. Such framework was also considered due to its usefulness to define areas at whatever scale the developers wanted to, provided they had a way to identify where that scene corresponded in the world. In the case that scene X sent an event, but the simulated scene X did not have a case to handle it, an error would be notified.

A practical case for this approach, which does not exclude using the scene as the area identifier, would be being able to define a street within a level, a city that is the entire area, and an entire country that has no model, based on the contextual needs of the developers.

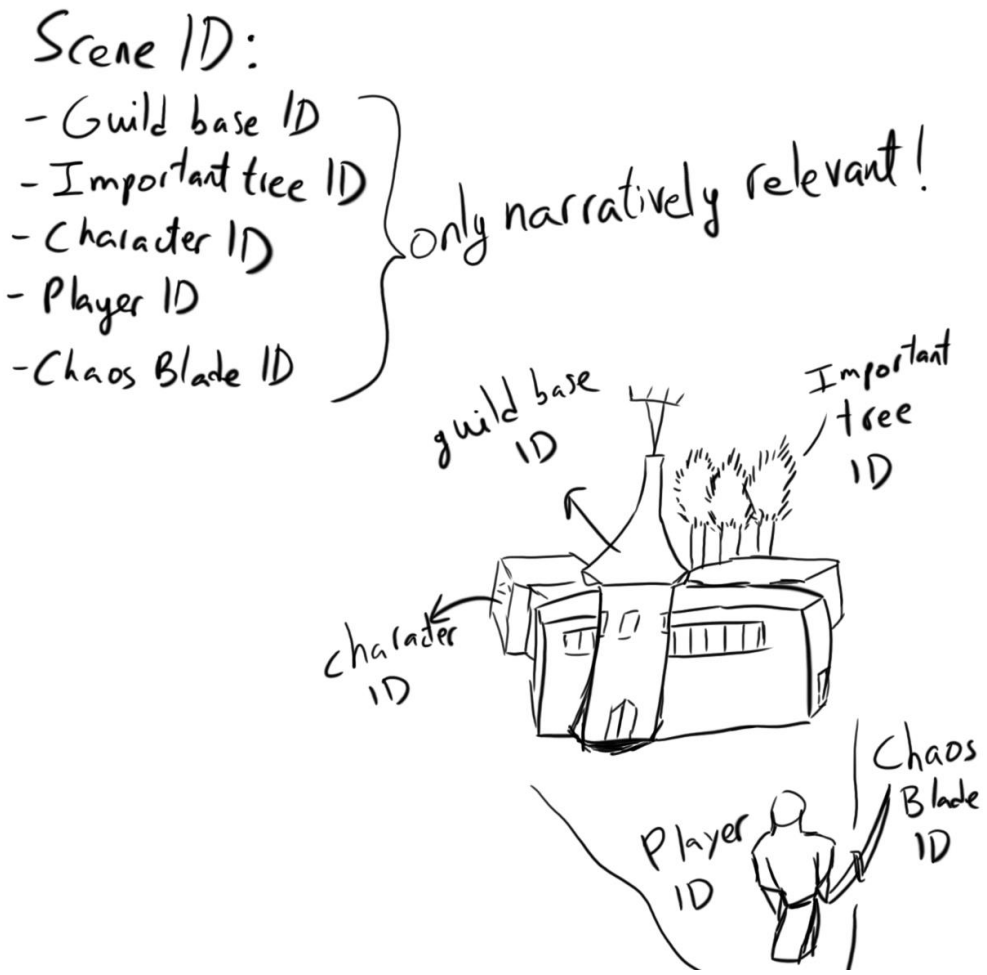
Figure 14 World simulation correlation with game world

The most important consideration that the narrative controller cannot account for is how the scenes are loaded: whenever it happens, the game should adapt all the elements that are tracked in the world simulator, and which are susceptible to have been altered while the scene was not being updated, clearing any possible discrepancy.

Those possible elements are divided into inanimate objects, characters and themes. Each element has a manually set ID that identifies them in each of the three "registers" they have: the game's true scene instantiation, that controls their current state; the world simulator's, which stores the ID in the proper cluster to easily iterate all elements of an area; and each element type's storage structure, which varies in complexity and autonomy depending on what it tracks,

and controls their changes of state based on narrative conditions. This ID should only be given to objects that will have a narrative function, to avoid unnecessary authorial burden.

Figure 15 World Simulator scene data storage



Whenever a scene has to be loaded, each narrative element instantiated (those susceptible of change) checks their narrative counterpart through their personal ID to see if they should still be in that scene. Conversely, when a scene is loaded, all objects that should be in it are sent to the scene, and if any is missing, we use the ID to know what has to be instantiated (the where is a matter left to developer hands).

While inanimate objects and themes' storing structures are within the world simulator, characters are found in the character controller, since their behavior is much more complex and closely related to narrative development to merit a separate system all together.

Such need for character autonomy forces a direct communication between world simulator and character controller for the sake of narrative coherence: when a character moves to an area more thematically adequate to the plot, they will have as their default knowledge (if non-other is

specified) the one stored in the world observer, and will use it to decide which and how long the best route to take is. Similarly, if the player has the option of fast-travelling, the world should step itself to match that period of time[56].

The method that will keep track of this connectivity and time per area travelled will be a simple dictionary that will store the origin as the first key, and a vector with all possible connections and their time as the second key. The same enum that was used for area identification would be used here.

A final note on area connectivity would be their typology in regards to the game scene:

- Current area: part of the world that is loaded, and the changes of which should be visually reflected if they are in the player's field of vision, with the coherence it entails.

- World area: part of the world that is unloaded, and which can hold non-visible or otherwise impossible events (for example, a contest took place, even if the game has no logic or assets to hold the player in one).

- Undefined area: taking a page from *Inform*, we defined a limbo where immaterial things are stored, also serving as a default value. This can be especially useful if the developers want to instantiate a known element that has narrative weight, but which is materially undefined at the start of the game, or if they want to have a character reside in a safe space the player cannot interact with until a more appropriate narrative context comes up.

With a solid grasp of the system's structures we can return to the problem of composing the world before the game starts loading maps. The truth is that there was not a good way to define an entry point for map information, since different map structures would have different better ways to be iterated through (it is not the same a game with one scene, one that generates locations procedurally, or one that would have to iterate across all its scenes to load them, check their content, and add them to the simulated world).

Since the ID standard was implemented, however, it would be enough to send an event to the world tracker from the game, of type "ADD_NEW_OBJECT" or so, and with the appropriate variables (like location and object type), and they will be permanently registered in the world.

[56] As a practical example, if fast-travel from X to Y would take an average of 40m, all the characters should be able to advance whatever journeys they may have partaken that same amount of time.

The only differentiation is if it is an area event, in which case we instantiate a new cluster; if it is an element, we initialize it adequately, and add it both to its containing area and its storing structure.

6.6. Theme control

Themes had always been the system's core feature from the start, yet in spite of it, they were never going to be a top-of-the-hierarchy, all-controlling module once the implementation began taking form. If there was a centralized system that dictated every theme interaction and forced instructions onto other modules' functionalities, we would be stepping onto the foundation of communication between system and player, losing emergency and getting a branched narrative system instead. Returning to its definition, emergency appears when autonomous agents take decisions.

From this conclusion it is not derived that themes should be casted away from the focus, but that they should adapt to system independence: as a result, they will have their own controller, while other modules have, within their instructions, thematic intentions.

To exemplify the function of theme control, and the limits of autonomous theming, we will take one of the most prominent category, characters, to show how those two parts interact. Each agent has events stored to perform under the right circumstances, each with a theme assigned. When the player acts with thematically-imbued mechanics, the appropriate thematically-imbued response will be ushered while checking consistency through the context (otherwise, it will be a mundane character response).

Justification for this approach is simple: abundant player agency implies that they have the possibility to disrupt any kind of thematic line the system might try to impose (unless everything revolved around a single theme), so coherence can only be achieved by having the world adapt to the ideas that player action introduces.

If characters, and other narrative elements, are thematically self-sufficient, though, the usage of the theme control system might seem to come into question. Nothing further from truth, there are still two major functionalities to take care of: theme definition and theme exploration. As an advanced summary, those functionalities are catered towards distinguishing player's developing themes and furthering theme discussion without ending up in redundancies, by

providing feedback and guidance to systems that make use of themes, not to directly control any specific part of the game.

6.6.1. Theme definition

Every narrative component, let it be for this case anything with multiple theme reactions defined within, will have to manage their theme stance (their perspective) once an event is completed ("learnt from"). Making this decision cannot be left entirely to the component, because it does not (and should not) have a sense of theme progression; otherwise, we could end with multiple characters essentially exploring the same idea independently, greatly damaging the final experience through redundancy, as it was seen in *Degrees of Lewdity*.

Theme definition, then, is a web of theme relationships that serves to express how themes can evolve. Each theme is defined as an enum, implying an intended sense of nebulosity, putting the emphasis on the container of an idea that relates to others, not so much on its meaning. This abstraction helps developers reach whatever granularity and variety fits their project, also solving the problem of distinguishing deterministic and open themes, since what makes them different is ignored by defining themes as enums.

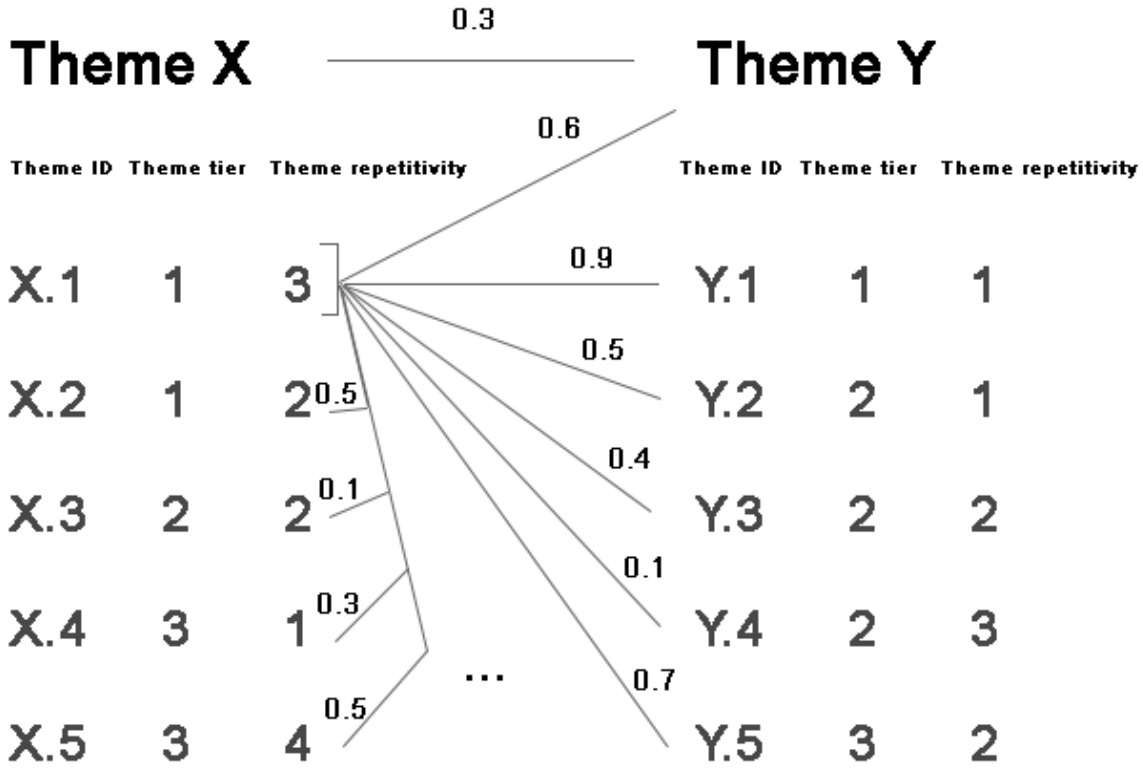
Relating themes is the core of the module, and their connections expand either vertically or horizontally:

- Vertical expansion consists of a list of different explorations of the same idea, divided by a tier value that separates the stages in which it is presented, and a weight value that defines interest between them; they provide inter-diversity, and allow for an easy implementation of open themes.

- Horizontal expansion, expanding on *Inform*'s behavior control, it's the definition of multiple, distinct ideas, which connect with each other through a series of weighted nodes, applying both for the macro-level themes, and the explorations within each. The weights define how interesting the developers consider each theme relation to be; when a component asks for guidance, we will be able to prioritize the most interesting response, similarly to most of the systems explored in the 5. State of art, but with a theme-based focus.

Mapped relationships will also have to account for possible dynamic changes: interest might increase under certain circumstances, and it may be better that some explorations were only reached if a certain idea had been presented previously. Both features can be fulfilled by changing the weight value when the proper event is received by the module.

Figure 16 Theme mapping structure



6.6.2. Theme exploration

Because the player has the active voice when selecting the themes, there might be the case that a component registered under theme X has to respond in a way that puts them into theme Y. If whichever system in charge concludes they are better suited to follow that path, they should have that theme assigned. In case the change was made, and then they evolved back into theme X, it would be out of place if they explored subthemes they (or other characters) had already been involved in.

A better experience would be conceived by avoiding repetition, and dynamically altering weights to favor choosing unexplored theme facets. Consequentially, a register should be kept to know what has already occurred, which theme lines are being occupied, and which can, potentially, bring more novelty (in the form of subthemes explored divided by total subthemes). It is relevant to consider that exploring a theme once might not be adequate for all cases, so all themes should have a value that defines how much they can be repeated before being considered sufficiently explored.

On top of that, whenever a component changes thematic lines, it would be desirable to avoid loss of progression, or repetition of ideas. Here is where the tier separation of subthemes comes into play: each tier identifies the component's level of experience within a theme. Eventually, if a change happens, a new subtheme for the component will be selected, choosing the one that relates the more closely to the current one, avoiding repetition[57]. When it happens, the component's tier level should descend to account for the theme's unfamiliarity, but not going to the lower level, as the new theme should not be distant enough that the component is "clueless" about their new situation.

The best way to achieve this would be for the developer to define how to revert to another theme's tier through each subtheme, but knowing the burden it could suppose, we could automatize the process by rounding an expression such as "new tier = tier of current theme – (1 – tier of current theme * weight of how related the new theme is to the current theme)", assuming tiers and weights were normalized between 0 and 1. Once computed, the system would return to normal functioning: any subtheme below the new tier would not be accessed, being considered meaningless or redundant. An exception would be made if the component returned to a previously explored theme, in which case their last tier would be directly assigned.

In summary, the system gets notified of either a component request or a component change regarding themes, and it responds by sending the most appropriate subtheme to follow on their next step.

Control of the player works similarly: registering the thematic actions they take, theme control identifies the subthemes they engage in, and handles their progression in the map by

[57] As a side note, it would be incorrect to think of characters for this part of the system, as characters, as we will see later on, do not have themes within themselves; they have story beats, they fulfill through goals, with a theme assigned to them. What would be thematically changing here, then, would be a particular story; if the character prioritized another goal, however, the character would not necessarily change (more of this will be specified on how characters are managed).

signaling to other systems how to correct their state into something more akin to the player's context.

6.7. Character control

Controlling characters is a complex task derived from their relation to both a thematically coherent plot, and their need to adequately react to their simulated, player-inhabited environment. To fit those roles, each character can be understood as an agent with a thematic goal that can vary based on the interactions lived on their way there. Ultimately, characters are stepped in phases: from defining an objective, to accomplishing it or failing to do so, to redefining the new goal. This process of decision-making and acting is defined by multiple functionalities.

6.7.1. Character manager

The character manager has all character objects registered, and is responsible for freezing and unfreezing them. This can occur when the player stops the game (opening a menu, for example), when the limit of on-going narratives is reached, when a character has to remain inactive until the player interacts with them, or when a character starts going from one point to another. For this last case, an in-game timer is assigned to the character, unfreezing once it runs out. Since one of its functions is character time management, it has access to the time functionality, and dictates all actions referent to that parameter.

Additionally, the character manager is where characters receive the event manager's signals from, and thus, functions as the primary communication channel between thematic control and narrative structure, and characters, accessing their parameters and goals through base class methods that issue the behavior correction request. Such communication can also go the other way around, requesting or informing other systems about character state.

6.7.2. World knowledge

A feature that recreates the world from the eyes of a character, adjusting to their perspective, but causing a deviation from reality, a feature also present in *Lume*, which provided more

narrative believability by limiting how much perfect knowledge characters had access to. It was more prominent on past iterations of the design, when there was the intention to have all interactions as parametrized values, including themes and ethos, but as that idea was quickly discarded due to immense authorial burden, it was relegated to providing coherence[58] and autonomy.

On the current state, world knowledge takes the world definition as a basic template, and adds a value of certainty to each world fact. Each character has a defined world by the start of the game, and it gets updated either through faction information, communication with characters they have a positive relationship with, world announcements (things that are *vox populi*) or their own experiences, each with different levels of certainty.

The major downside of this part of the system is its lack of story support (Ryan, Mateas, & Wardrip-Fruin, 2015), not because it cannot be implemented, but because doing so would limit the possibilities of an extradiegetic-interaction-free game. It is, then, left to developers' hands how they want to contextualize feedback that communicates the transmission of communication to the player.

6.7.3. Factions

Factions define any group a character is part of, and from which it is willing to take orders; if such is the case, the character has an allegiance to that faction. The only limit of being part of a faction is to not be part of an opposing faction, unless it is the character's false faction, which simulates espionage dynamics, already seen to work successfully in *Fallout*, providing good narrative value and dimension to a relatively easy-to-implement feature.

Faction relationships are managed like characters, from the character manager, but in a different container. As any other component, it contains thematic elements, which serve to enhance the narrative, to coordinate multiple characters under a story event, and to dispel allegiance in case of a thematic backlash (representing a dissonance between the character and the faction's ideals).

The feature is managed based on agent action: every time a character, be it part of the faction or not, acts, the faction registers how it affects them, and how they should be viewed by its

[58] It was especially important for characters that existed isolated from the world, or to reflect its vastness, as they would start with a relatively small world, and expand their action space as they ventured further in it.

members in response. The parameters to be used were considered to be multiple axis (like helpful – detrimental, controllable – free…) that are manually altered based on the registered action. Simple parametrization was deliberately avoided to allow a character to inhabit many factions, as it was expected that having a single good – bad axis, even if it had tier levels, would end up being very reductionist.

On addition to the nature of the action being registered, we also wanted to simulate the bias effect of social groups, as mentioned in Ken Levine's talk (Levine, 2014), as a conditioning element to faction reputation. Source believability was also a desired factor to be considered. Considering those elements, an approximation to calculating action reputation would be, for each agent reputation axis, "10%[59] current reputation on that axis + trust on source * action's intrinsic parameter scores".

To avoid minor actions to trigger major faction repercussions because a certain threshold was passed, the system would implement a tier system that required a minimal parameter augmentation (or decrease) to reach the next level, taking the pragmatic approach that *Charade* adapted.

Once the faction receives new world information, it should communicate it to all its members. This can be done automatically (for narratives with settings that have instantaneous communication), or through the same algorithm of map traversal (to simulate word of mouth). For the second option, it is advised to use different weights on the nodes, because they should not represent distance, but the faction's effectiveness to communicate. Proper handling of this functionality could lead, for example, to situations where the player interacts with members of a faction who are not aware of the player's reputation, but which might be informed of it at any moment, creating an emergent player dilemma.

One feature of the system that was decided to be excluded was faction inter-hierarchy, the internal relationships that define who controls the faction, and any possible conflict that stems from it. While it could be used for thematic effect, it was deemed better to have developers code the event themselves, instead of having a system, because it would most likely overload the world complexity, and generate noise on the player's perception of the plot.

[59] Illustrative value.

6.7.4. Relation mapping

Relation mapping serves as another source of information for decision making, determining the relationships between a character and all its acquaintances. Similarly to factions, relationships are parametrized through ranges on independent axis, divided by tiers; unlike those, however, they are affected by the character's personality, represented in multiple ways. Alike approaches were taken due to the multiple facets they allowed developers to represent, and how positive that variety is to player experience, as noted during the *Degrees of Lewdity*'s analysis.

Firstly, the axis parameters have points of no return that will lock a relation on a certain range, to avoid crucial thematic stances to be reversed. On a similar note, different characters will have different thresholds for each tier, manually introduced by the author or inherited from their base class. Additionally, there will be grudges, events that occurred in the past that will lock a character on a specific range of the relationship until a certain amount of conditions are fulfilled. Many of those features were present in multiple analyzed systems, such as *Inform7* or *Charade*, and the self-evident utility they provide was enough to quickly incorporate them on the dynamics of character simulation.

Finally, each character will have a series of multipliers to compute in the formulas that decide whether a character prioritizes a behavior or not. They can be unique to them or common to the base character class, like faction submissiveness, or whichever the developers wanted to add.

Every time the character manager sends an interaction with a character, the relation map is checked: if there was a previously established relation, it applies a bias on the current relationship in whichever axis the action affects, plus the inherent values of the action.

It was originally planned to have relations tend to neutrality if nothing meaningful happened between two characters, in the same way in which *Degrees of Lewdity* treats stat progression, or *Charade* does with character progression. It was ultimately deemed an unnecessary feature that could trample over thematically important set-ups, and which could be hard for the player to wrap their heads around, intruding in the game's basic loops, unnecessarily sacrificing the quality of the experience for a tinge of realism[60].

[60] Realism that was not a priority from the beginning (not to confuse realism with believability).

6.7.5. Agent action

Character actions can come from two places: as a reaction or as an active action. Reactions are triggered when someone or something acts upon the agent; those are usually the same for every character, and are sampled based on which mechanics they have available. Once they are done, they notify character manager of any theme change, and generate a new goal if necessary.

One thing worth noting about reactions are that we will not make any consideration on sensorial detection, as games like *Fallout* do, because that is task left to the game's systems: we only take the event that it has been detected, and proceed accordingly.

Active actions, on the other hand, are a bit more complex. They are designed to fulfill character objectives, and are based on an HTN planning system: when an event takes place, a goal might be inputted into the planning structure in the form of desired values for the character's world knowledge state. From it, a sequence of available actions that fulfill the goal is returned, dictating the different behaviors the character has to follow to succeed. Not all characters will have the same actions, and not all actions are limited to those available to the player: this freedom, exemplified in both *F.E.A.R.*'s and *Charade*'s action sets and *WAWLT*'s non-available actions, easily amplifies the developer's story-telling reach without increasing the development burden.

Each character will have, most of the times, a series of goals to fulfill, each with a priority assigned, which combined with their thematic priority, returns the value from which an objective is chosen. Priorities are separated between numerical values, top priority (highest non-mandatory priority) and guaranteed priority (mandatory; the system will execute the first available).

At every time, only one goal can be in progress of completion, to avoid abnormal behavior, and every time the world state changes, the sequence of events to reach success will be recalculated (and the priority might change). Interrupting a goal sequence will only happen to trigger an agent reaction, or after a unit of the sequence is completed, helping avoid uncanny behaviors, following what was implemented in *Façade*.

If a new goal is established, the prior sequence of events will be revised: if completing is on its last steps, and its priority is lesser than a given value, the character will finish it even if there is a higher goal. Once completed, failed, or with the goal successfully changed, the system will re-run the planning system until a new solution is found, or will attempt the last chain of actions

tested, for a defined amount of times, before changing goal priority if there are no new possibilities to attempt.

HTN planning was implemented over a FAtiMA-based system mainly because we could not depend on a parametrized world to develop the narrative, and instead had to rely on manually authored content, at least partially, to generate dynamic narratives.

6.7.6. Behaviors

Behaviors are the smallest unit that compose character agency, and the result of all the above features interaction, as well as a core part of our compositional representational strategies to decrease authorial burden (Ryan, Mateas, & Wardrip-Fruin, 2015), as it was seen in *Façade* and *F.E.A.R.* When an action is chosen from the planning system's results, the agent enters the correspondent function, containing any necessary check to return a "success", "on-going" or "failure" result.

Those checks can be made on the level of any of the previously mentioned character features, but the agent will also contain a dictionary variable, at base character class level, with a list of events and a struct containing an integer[61] to mark whether they have occurred or not. This gives them the freedom to register any event dynamically, while providing a very generic, useful structure to check information from, that is not contingent to the event's content. The mentioned struct also contains narrative-centric data of the event, such as its pace or tone.

Once all checks are made, if a behavior has to notify its in-game counterpart to change state, it will check the agent is loaded and send it; otherwise, the agent is out of the loaded scene, and the behavior is stepped automatically. Once a behavior is completed, it sends a signal to the character manager to deliver it to other systems.

6.7.7. Joint actions

Joint actions are a special type of character interaction that takes place when two characters with the same goal with a positive affinity meet (although developers could invoke them even if there was a negative relation). They are directly inspired by *Façade*'s own version of joint

[61] Generally, we would only need 0 or 1 (bool) to check whether the event took place or not, but in order to give more freedom to the developers, they can register any state of the event with a series of ints instead.

actions and *F.E.A.R.*'s group coordination, a feature that expands the developer's thematic range by allowing entity coordination.

When a joint action is initialized, both characters will get the generated goal as a priority, which can include special, coordinated behavior. Its management is taken care of by the joint action, which also tracks character availability; if one of them becomes unable to keep their part, be it by death, inability or a change of goal, the joint action will reduce the affinity between both characters based on the priority of the event. If it succeeds it will do the opposite. In either case, once the action is concluded, the joint action is deleted and character recover their full autonomy.

This aspect of joint actions directly mirrors both the retained individuality present in *F.E.A.R.*, something that only makes sense when representing self-aware entities, and *Charade*'s social consequences, which adds meaningful aftermaths to breaking an accord.

6.7.8. Decision-making

With all the above under consideration, every decision takes multiple values: their relationships to the agents involved, the faction affinity, the certainty of the information they are basing their actions off, the potential gain of succeeding, etc. However, none of these factors are implied on every decision, as it will be the developer who will directly define what they want to consider on each choice.

The conjunction of all the factors that influence decision-making tries to be a mix of social and individual forces within the character, in a way resembling *STellA*'s character layers, to allow enough diversity of action (and reaction) to explore distinct facets of the human experience within the thematic context of the story.

6.8. Narrative structure

Narrative structure was originally postulated as both a controller of plot events' order and timing, and as the manager that defined which beats would be selected to continue a started narrative thread. Its functioning was exemplified with *PropperWryter*'s implementation (Gervás, Concepción, León, Méndez, & Delatorre, 2019, p. 2), and was a procedure shared by

narrative generators such as *Façade*, *Lume*, *DINAH* or *StoryAssembler*, or mixed in the procedural nature of games like *Road 96*.

A system that defined plot order through templates or beats, however, directly contradicts our agency-centric approach: at this point in development, we had a theoretical world the player interacts with in whichever way they want, and they get a response by autonomous characters with their own goals. If we were to predefine plot beats, characters would be tied to predefined actions the consequences of which would hardly be expandable outside of the beat, and character continuity would have to be manually built.

With the intention to keep this kind of narrative set-up, plot constructors that were based on template beats were mostly disregarded, as well as other approaches like *Encounter Manager*'s goal-directed plot building.

It is important to point out that, at a theoretical level, it should be able to combine plot imposition with character autonomy, and our system tries to simulate this effect, but an implementation of both full-fledged systems escapes the scope of this project.

Prioritizing our approach to the game experience, then, we have to define what our narrative structure will be taking care of.

6.8.1. Plot regulator

Plot regulator is the subsystem dedicated to blocking or allowing the progression of every present narrative thread. Its most complex task is to regulate which characters that are ready to have their plots progressed are given permission to enter the same space occupied by the player.

Every time a narrative progresses, the plot regulator should receive the correspondent event and register some basic questions of a story: who (with the character id), where (location enums of the world observer), what (event id), why (theme id) and when (game clock time). The regulator should also have access to which characters are available to have narrative interactivity.

Registration through event is our system's version of narrative memory, which we opted to track extensively, following *WAWLT*'s approach instead of other simpler recordings, like *Lume*'s, as a consequence of the open-ended type of events we wanted to implement: if anything can be made into an event, we ought to be prepared to at least hold some residual information that gives insight into the context of that moment, for the sake of future coherence.

Knowing the event details and the available characters, a formula can be devised to choose the best next option/s to present to the player. As with many structural features, the system will propose a basic approach, but it is incentivized that developers adjusts it to their needs. An example for this case would be using different formulas for different narrative moments. As a starting point, however, we limit our calculations to:

1. Pace, tone and emotionality. Common parameters for most narrative manager, they will be given a bit of a special treatment to fit in our system. All three parameters are computed separately by using the same logic: the system has a register of the highest and lowest value for each axis; it calculates the difference between them, and undergoes a series of checks; depending on the magnitude of the result, and the previous sequences, a value will be returned. As much as this mono-dimensional approach can be easy to understand for pace (fast – slow) and emotionality (high – low emotions), tone can be a bit more difficult to define (dark – light topics, serious – comedic presentation, dark – bright visuals…). Nonetheless, it was decided to follow this pipeline, since it was supposed to be but a guide for more specific developer-coded algorithms.

2. Goal success. Similarly to STellA, player progression will be used to control pace, in the form of a value that is returned based on how long it was since the player achieved an objective. The longest the time, the higher the value, thus the more score will be gained by fragments that are identified as close to reaching a conclusion (which means a possible goal success).

3. Priority score. In *The Long Path to Narrative Generation* publication (Gervás, Concepción, León, Méndez, & Delatorre, 2019), a system named "information reveal" is considered, to manage not the frequency of player-received information, but the quality of it. Our more agency-centric take complicates this task, because we are leaving the character stories to their own (and the player's) hands. An attempt to cope with this dichotomy, even if in a very limited way, is the "priority score" value, which codifies an event as more or less important to be continued after it is done. Any specifics on information reveal, then, are handled within the event that continues the prior occurrences.

This take is directly inspired by *Façade*'s narrative construction, as the dynamic granularity it offered to take informed decision was a powerful feature to regulate plot.

In addition to the already mentioned functioning, there are two extra considerations to be done: specific events can be made to have specific follow-ups for all categories discussed above, and also, a strike system will be implemented to avoid stagnancy. Normally, the plot regulator would check the values of all available plots, and compare it to a number: the top scoring result will be picked as the next event, unless it does not fit a minimum value criterion. Strikes count a fault every time there is a consult to advance any plot, and there is no answer to fulfill the narrative condition currently in; if this happens the values of the previous event will be scaled down a bit. If by the third attempt there is no available option, this minimum value will be ignored, and the best plot will be fired into the game's world.

Again, it is advised to customize the narrative algorithms, for limiting the storytelling to a single progression template can imbue the narrative with a sense of monotony and predictability that work against the narrative immersion necessary to make meaningful decisions. This is true for videogames, and not so much for static media, because in games, players are forced to take decisions; if the form of the artwork becomes a clue to solving a problem, it becomes a factor that conditions player choice, thus hinders the process of introspection.

6.8.2. Pattern recognition

As a consequence of storing narrative events, the system is capable of detecting narrative trends by defining a series of conditions that would be checked using the plot regulator's narrative memory as a database, combined with other tools like time control (for example, to check that something happens within a certain cadence).

Pattern recognition is an adequate fit to fulfill the input complexity goal we established at the beginning of this research, something that can be seen put in practice in *Middle-Earth: Shadow of Mordor*.

6.8.3 Forced narrative structure

Whilst a character-centric approach is what we thought would work better, ignoring narrative structures as a tool to be used in the system would be a poor decision; after all, the problem is in their abuse for constant composition, not their utility or application. Consequently, plots that fitted certain types of structure, or that were predefined to follow a certain one (for example,

central plots that would be more desirable to keep a tight internal pacing on), should be able to have assigned a specific narrative structure so that the system executes it as the player progresses into the world.

To do so, using templates that are filled in-run with the agents that participate in them would give us both the control and flexibility we needed. While the template approach is something quite extended, as seen in the cases of study mentioned in this section's introduction, we focused on something more plot-centric, like the *PropperWryter* system, rather than using them for characters, as we already defined an inheritance structure to provide common character roles.

A starting point to implement this feature was the CYOA narrative progression typology (Ashwell, 2015), which gave many content to begin working with; this is not to say that those cycles would not be subject to the plot regulator's control, but it would facilitate character behavior, making it a generalist plot thread instead of enclosed within the character, or decided by its goals.

In summary, this would be a part of the system aimed at assigning characters an action roadmap through theme correlation, without having to specify who would fit in them, achieving a certain goal independence, with the possibility of assigning those structures to any character at any point, disregarding partially or fully their background to fulfill a particular narrative role.

6.8.4. Plot manager

To enable the plot regulator's functioning it is required to have a feature in charge of managing the moment to moment before any narrative request is made: introducing the plot manager, a glorified container for registering plot threads. This section does not explain any new functionality, just how narrative structure is composed in the inside: each thread would be an array of plot events identified by the character ID, functioning as the system's narrative memory handler, registered whenever the narrative structure system received a signal from the event manager. If there was no ID in which to push the new event, a new array would be created.

In aims of improving the underwhelming story support and story recognition factors of our generalist system (Ryan, Mateas, & Wardrip-Fruin, 2015), options were included to regulate the amount of plots that could be taking place at the same time (including those occurring outside the currently loaded scene) and the number of plots that could be interacted within the

same space. Whenever this number would be trespassed, or dynamically altered in run-time, the system would freeze the lower priority character threads (a mix of attachment to the character and thematic uniqueness); conversely, this check would also allow the acceptance (or issuing) of narrative progression requests.

Prioritizing player-relevant plots instead of thematically-relevant plots was a decision taken in favor of reducing player frustration and improving engagement, an approach that seemed the most reasonable, and which was adopted too by *Degrees of Lewdity*, *Road 96* or *Middle-Earth: Shadow of Mordor*, and which resembles *Hades*' dialogue system in terms of internal structure.

The plot manger should also have a bit of narrative control; in the same way that characters can store flags to control their behavior, or the world observer can store variables that register past states of the world, plot manager can save information to trigger certain events whenever the narrative conditions are proper. Very related to this feature is zone unlocking: since the player will most likely not have immediate access to all the map, developers can choose to limit story progression to those that occur in certain spaces, and unlock barriers as the player progresses.

6.8.5. Plot detection (deprecated)

Plot detection would have been a sub-system dedicated to use the pattern recognition to identify possible thematically relevant plots from player action, and develop those beats into full-fledged narratives. In the end, the impossibility to delegate deep thematic recognition and development to algorithmic functions made this system redundant, as "detecting" a plot would ultimately mean that a plot had already been triggered, and thus, the system was already aware of it.

In case that the prior problem was solved, and this system gained relevance again, we based most of our research onto Genevieve Lively's narrative insight about the art of anticipation to define plot detection conditions (Lively, 2017): a basic model we proposed would be free-form story making, where we take events pre-defined as relevant, and connect their participants, indicating to the narrative structure that those should be involved into a future narrative. Because humans tend to connect events as stories, the model bypasses context limitations, focusing on providing a ground from which a series of events can be built, and because a character affinity system should be going on in the background, coherence problems should be avoided. Because this approach is quite simple, though, it lacks the desired depth.

Another approach that can remedy that is through foreshadowing: if certain conditions are fulfilled, a pre-narrative condition is checked, which is followed by a pre-designed narrative mold, continuing that initial interaction. Even with the implication of authored content, if narrative structure can (as it should) be codified in simple instructions, which the actors can fill independently of who they are, the cost should not be comparable to the freedom it could grant. This take, being more static too, mainly fails in its lack of thematic consideration, which was already stablished as a project goal. To resolve that problem, two approaches were defined:

Thematic abstraction. Because a world state would be tracked (or be accessible, at the very least), we could abstract situations in which multiple themes were in contact, to define character behavior patterns that could generate chains of reactions, eventually forming a concluded plot. To do this, a theme bible would be required (somewhere to define the signifiers of each themes and the relationships between them), as well as the system to make the context check, and the character behaviors that could thematically respond. However, because the system aims at being automatically dynamic, without having to rely on authored content, what could be done would be changing the characters defined values to raise the probability that a certain type of story would unfold. This approach seems not only effective at its objectives, but coherent too, as long as this is done with characters that have had no connection with the player previously, otherwise the personality override would break the immersion. Nonetheless, this approach still requires a theme bible, which adds a factor of uncertainty to how viable it is to develop.

Anticipation. The other approach would be to define the occurrence of an event, and have the dynamic stories spurt as a reaction to that prediction, a take that aligns with the storytelling generative system author Natalie Dehn (Gervás, 2009). This facilitates theme handling, because everything can be made to spin around the same idea (or cluster of ideas), and all behaviors can be planned to work under a set of specific circumstances, while not growing authorial burden exponentially. This reduction of character response range, on the other hand, comes at the cost of lesser variety in every narrative set, and even as a restriction of creative freedom. It is also not easy to assert, at a theoretical level, whether multiple narratives sets could be concatenated in the same game world consecutively, as the dramatic change of thematic rules could create inconsistent behaviors.

6.8.6. Information reveal (deprecated)

Despite being one of the pillars established in *The Long Path to Narrative Generation*, a system that controlled the rhythm and quality of the information being revealed was not ultimately implemented, as a result of the system's open-ended nature (a generic structure could hardly be used to very differentiated types of games), and its higher focus on thematic construction and relation rather than engagement values.

6.9. Time control

Time was considered an important factor on keeping internal consistency, but contrasting with *DINAH*'s approach of separating time in different categories, we would take a more flexible approach. Within a time-dedicated module, there would be an array of timers that would be running based on a time scale, with a time limit (as if it was a countdown) and the event it was set-up from. Once the timer had gone to 0, a "TIMER_EXPIRED" event would be sent with the triggering event as a variable, so that whoever sent it in the first place could identify the appropriate response.

As it can be seen, this is quite a simple module, that only requires three more considerations to be made: firstly, about time scale. Each diegesis usually wants to run time in its own pace: games like *Pokémon* have seen both a daytime adjusted to real-world time, and relatively fast iterations to allow constant access to the different animals of each zone. To allow for maximum flexibility, a time scale variable will be kept to convert real time to diegetic time; on top of that, multiple types of conversions should be manageable, as it should not have to be necessarily the same time scale when exploring the world than when talking with an NPC, in a fight, or when shopping in a town.

The second concept to be explained is that of teleporting or zone loading – unloading: whenever such phenomena occur, a value should be sent to update time adequately, not from a general map that calculates the most efficient theoretical time, but from the agent of the event using their own (faked) world knowledge.

Finally, timers should be able to be paused and reactivated whenever the proper event is sent: a "STOP_TIMER" or "REACTIVATE_TIMER" event with the event of the timer it wants to alter. This process could be in itself taken care with a timer (it would be ill-advised to concatenate too many of those).

6.10. Data saving and loading system

The purpose of this system is to save and load all the appropriate data to regain state every time the player loads a game. For every system that required this functionality, an inherited pair of functions would be available, and each would generate a .txt file with the necessary data, all saved in the proper directory. To do so, we used *PhysFS* and *PugiXML*.

6.11. Debugging system

Dynamic narrative control based on simulations' rules implies permutations of world states hard to grasp for the human mind; the complexity of multiple agents acting independently easily generates results the causality of which is superficially unclear; major interrelationships between subsystems can scale relatively minor bugs into inscrutable results, etc. An ominously long list of such possible problematics derived from the system's nature quickly justified the implementation of a debug system, to monitor beyond the practice of debugging itself.

Ensuring the correct functioning of the system was obviously important, but being able to develop structures that test its state concurrent to its development was even more crucial, because as the code grew in complexity, it would allow us to pinpoint errors based on their origin in sub-system interaction, or in sub-system implementation.

The debug system was not only relevant to the project, but to future developers using the tool, since there will be authored content that has to be fed to the system. As a result, it had to provide intuitive affordances that facilitated its usage.

It was clear from the start that the system should reach every other sub-system, which presented two necessities that had to be accounted for:

1. Required perfect information of what was occurring inside each sub-system.
2. Had to be able to retrieve and adapt to store that information from the different structures it took it from.

Incidentally, as it was rightly pointed out when discussing their Intelligent Narrative Feedback approach (Koenitz, Ferri, Haahr, Sezen, & Sezen, 2017), the debug system should provide information in real time to see data management as a direct consequence of player action. This

led us to stablishing multiple debugging modes, all of which should work independently, but be available simultaneously, as well as having available which data to display.

System mode:

- Action feedback: informs about player action taken (or attempted) and its consequences.
- World feedback: informs about initial state and changes of the world.
- Character feedback: informs about character state.
- Structure feedback: informs about narrative progression.
- Theme feedback: informs about theme progression.

Input mode:

- Player input: application combines player action with world events to generate data in real time.
- Simulated input: taking a page from *Encounter Manager*'s iterative process, the application generates player action randomly or following certain instructions (usually useful to test extremist action on either side of the mechanical spectrum), stepping the diegetic time as fast as possible.
- Frozen input: application computes the world state at maximum stepping speed, as if player remained inactive.

Output mode:

- Run-time feedback: all information is delivered while the application is running, in an appropriate format that places special importance to being easily understandable.
- Registered feedback: run-time application is registered into a set of files once the application is closed or the game is rerun. Multiple sessions should be stored, and all the information should be distributed among different files for easy access. As a consequence, the storage of the data will follow a hierarchy of a folder named with the time at which the session begun, and when it ended, inside of which all the necessary files would be found.
- Dual feedback: uses both run-time and registered feedback.

Event mode:

- Routine checks: all information that is expected and coherent behavior; if any manually specified combination of factors is detected, it will be automatically reported.

- Meaningful action: all changes in the world that represent intense emotional beats or game-changers for the world state.

To keep track of most of the information discussed above, we decided to rely on the Manager Manager, since event communication, as useful at is to make the system work, is not a representation of the world state, only of what occurs in it. While it is a factor that will also be monitored, the figurative screenshot taken of the world will be directly extracted from the modules, and stored (if the debug mode so requires) in .txt files through *PugiXML* and *PhysFS*.

6.12. Manager manager

The final piece of the system lay in having all those sub-systems instantiated and controlled at a high level, as our objective of system independence for the sake of avoiding code dependencies did not exclude hierarchical control.

Its most important functionality is storing the event manager: the manager manager would be a static, pseudo-global variable with a public function to call event registration from anywhere. Once registered, the event module would check which systems had been instantiated within the general manager, and would proceed as previously explained.

The second feature of the system is communicating internal information to the game's code: when the developers have their own update loop structure, they can either directly access the insides of our tool, or rely on the information communication system to receive updates of the internal state of the narrative to handle their own features.

In this regard, every event (which are the most generic, extended and accurate type of information within the tool, since everything is internally communicated by them) will be notified to whichever receiver the manager manager is pointed towards. Additionally, any variable of any sub-system that can be accessed is susceptible to have a getter done to retrieve it to the outside, albeit those functions would have to be created by the developers. Nonetheless, some basic ones were implemented, as they served fundamental parts of the tool's structures.

6.13. Technical discussion conclusions

With all the details and decisions explained, it would be time to put the system's final design into perspective. Holistically, it can be understood as a tool for dynamic narrative control based on themes, one that encourages the developers to tinker and adjust to their needs, in opposition to traditional libraries, that provide a functionality to be used. It could, then, be defined as a structural template.

Internally, the system receives narratively relevant elements from the game world, and through a series of manually set events, certain actions take place. Actions can be manually adjusted, or have a generic, algorithmic response, but in either case, they depend on a narrative controller to be executed, which on turn consults a constantly updated, manually set, theme library to formulate its decisions. The back and forth between player – system input – response and vice versa constructs the story, which can, at any point, be tracked externally through the multiple debugging options offered.

The implications of the restrains around theme expression meant that we could not have dynamic generation of content, but the management of the semi-predefined events has enough room to consider the system a completely dynamic narrative manager, which was the objective all along.

7. Technical demo

With the system's design intricacies solved at high level, and a planning developed to implement it, it would be time to begin defining the specifics of the final results, and what value it can give to users.

From now on, we will be referring to the developed product, and thus, not talking about an abstract system, but a concrete creation. As such, it was decided that it would be more clear if it was given a name, and to that end, we baptized the tool as *Yume*. Complementing said tool, we developed a technical demo, that was named *Yacchatana*, where we showcase *Yume*'s capabilities, and which can be used by developers as a practical guide.

It is fundamental to understand, however, that they are two very different things, as the usage of *Yume* requires a considerable adaptation of the base code to the story it is being used in; consequently, *Yacchatana* is a far cry from how other games that use *Yume* might look and internally work like.

7.1. Yume

This segment will be dedicated to unraveling *Yume* to its core pieces, attempting to provide a handy but complete introduction to the tool, while differentiating it from the theoretical approach that was proposed in the 6. Technical discussion.

On the surface, a game *Yume* is implemented in has to worry only about staying in contact with the Manager Manager. Once instantiated, its Update method is called; additionally, a PreUpdate and PostUpdate methods were included just in case, although they are currently not being used.

```
while (mainLoop == true) {

    SDL_Event e;
    while (SDL_PollEvent(&e))
    {
        ImGui_ImplSDL2_ProcessEvent(&e);
        switch (e.type) {
        case SDL_QUIT: mainLoop = false; break;
        }
    }

    ImGuiStartFrame();

    yume.PreUpdate(); // Any check we want to do pre-updates

    // Game "world" update
    yacchatana.Update();
    ManageYacchatanaEvents(yacchatana, yume);

    yume.Update(); // Yume update
    yume.PostUpdate(); // Any check we want to do post-update
    if (yume.updateScenes) {
        yacchatana.UpdateScene(yume.GetLoadedScene(yacchatana.GetCurrentScene()));
        yume.updateScenes = false;
    }

    ImGui::EndFrame();
    RenderPostUpdate();

}

yacchatana.CleanUp();
yume.CleanUp();
```

SDL event check

Where the game would call its updates

Debug update

Figure 17 Yume instantiation

As it can be seen in the figure above, running *Yume* is fairly simple. Beyond simulations, however, the system also is responsible to communicate its state with the game, and to receive theirs too. ManageYacchatanaEvents is in charge to sending the game's events to *Yume* (again, in an actual game, this would probably be done by its own event manager, as this function is disposable and only for the purpose of showcase); on the other hand, in the "Debug update" section, we would have a function that takes input from *Yume*, and sends it out. Since our level of implementation did not require it, we simply added the "update scenes" case to show how it would work.

Once initialized and connected with the game, *Yume* will call each of the different modules and execute the proper functionality to simulate the narrative state of the game.

7.1.1. Module

All centralizing sub-systems of *Yume* inherit from a Module class that provides save – load virtual functions[62], and a reference to a static event manager. Making it static seemed the best option, because a global variable accessible from the game could cause inconsistencies, but at the same, all sub-systems had to share the same event manager.

Finally, in regards to the save – load system, currently not all modules save and load all data (mostly it is the ones that hold the world state, like Character Control and World Simulator), but they all have the basic structures to generate the notepad files to hold information, so the user only has to define which variables to save at their convenience. The modules that do save information can be used as reference to how data can be worked with in XML format.

7.1.2. Manager Manager

One of the main notable points of change with the theoretical implementation was that Manager Manager ended up absorbing the Debug Center system. This change was easily justified, since Debug Center needed information from the rest of modules, and the only way to get it would have been to request it from the Manager Manager. It would have been redundant to make it a module of its own, when it would be completely dependent, so they were, logically, meshed together.

Additionally, Debug Center did not have to store information (any debug output to be saved would be performed by the modules themselves, as they were the information containers), so not inheriting from the Module class would not require function re-definition.

Knowing the ImGUI debug code could take a lot of space, however, Debug Center was written in a .cpp of its own (although as mentioned, the functions themselves are declared inside Manager Manager class), to facilitate readability. On an additional note about ImGUI, there were doubts on how much we should expect our user to deal with it, since it has its own initialization requirements, which are technically not necessary to run *Yume*. Since most, if not all, ImGUI code is in Debug Center (so it's easy to comment), and ImGUI itself can be initialized with many render set-ups, we opted to only hold ImGUI functionality, leaving the

[62] In summary, that they have to be re-written and adapted to whatever the module needs.

initialization and window render creations outside *Yume*, at the user's convenience. Nonetheless, *Yacchatana* is still handling it, in case any inexperienced user does not feel confident implementing it by themselves.

In a way, Manager Manager also absorbed event manager: while it remained independent and accessible to other modules, it would be this one the responsible for, at every frame, getting all stored events and sending them to each module.

Finally, there was considerable concern on how to work with the modules that are instantiated in the Manager: one on hand, we did not want to have to iterate the storing vector every time we wanted to access a particular one, but on the other hand, we did not want either one pointer per each module, nor a hard-coded module order. The solution to this has been to make an enumeration variable that serves as an array's index: to access CHARACTER_CONTROL (enumeration value 0), we could make a cast:

```
(CharacterControl*)moduleVec[(int)CHARACTER_CONTROL]
```

And after making a check at the constructor that the enumeration value and the array declaration order corresponded, we could have the certainty that this style of casting was save for the rest of the compilation.

7.1.3. Character control

Character control did not receive many changes from how it was originally planned, although there are serious improvements to be made in future versions, as we will discuss later on. Currently, characters are a single, base class, in comparison to the levels of inheritance that were initially proposed: after all, while it can be easier to understand information separated in different classes, all the functionality could be held in a single one and still work perfectly well. Additionally, each character does not have its own class anymore, since it would increase authorial burden and compilation time without any benefit.

In the theoretical proposal, we also talked about how we would base our character control on an HTN planning system. Because character agency is a fundamental part of the system, we will go in-depth through the process they follow to act:

Characters have goals, identified by a GOAL_TYPE enumeration. Once a goal is decided, by checking their priority (a float), their ID is sent to the GetActionPlan function, where all

possible routes to achieve the goal are checked. The first one that can be performed will return a vector of actions necessary to reach that goal. Whenever the roadmap has been defined, the first action executes, checking again that it is possible to perform. Once it is done, the next action triggers, until there is none left, and the goal is succeeded.

If an action cannot be performed by the time it has to be executed, or even fails, a new roadmap is attempted. If that's also not possible, goal is considered failed, which has a special state to trigger narrative continuation by adding a goal to the character's pool if necessary.

At any moment, the character might get a reaction, which triggers a new array of actions, but does not erase the previous goal, unlike goal completion or failure. Similarly, we can also get a "reactive goal" at any point: they are a special kind, that if they had the highest goal priority, they would interrupt the current goal, but deleting it only if it was a reaction (it would not make sense to keep it if the context had changed), and being sent immediately to GetActionPlan. If it's not the highest priority, it simply gets added to the goal list. There is an additional check to consider: if the previous goal was of top or guaranteed priority, and had only one step left, it would be completed regardless, in an attempt to simulate humans' tendency to achieve closure.

Performing an action may be filtered by personality or relationship checks, two factors that have been considerably changed, too, from how they were conceived. Personality is composed by a set of values that can be used as multipliers when updating a relationship's status. Relationships, on the meanwhile, have a set of stats, the combination of which defines the different types of relationship from a character to another. A function can be called to get the different "facets" of said relationship.

```
RELATION_TYPE Character::IsEnemy(SCENE_ID characterID) {

    RelationSubParameter& relation = relationComponents[characterID];
    if (relation.trust.value < ENEMY_TRUST && relation.yearning.value < ENEMY_YEARNING && relation.indebtedness.value < ENEMY_INDEBTEDNESS &&
        relation.inconditionality.value < ENEMY_INCONDITIONALITY) {
        return RELATION_TYPE::ENEMIES;
    }
    return RELATION_TYPE::UNDEFINED_RELATIONSHIP;

}
```

Figure 18 Relationship checks of the "enemy" facet

Such an approach allows for a clear way to obtain information, while not rejecting the variety of facets a relation would naturally have in reality. To distinguish the importance of said facets, a tier system was implemented. Unfortunately, the limitations of the entire system only allow for a linear increase to go from a tier to the next one, but personality values can be used to

regulate relationship growth nonetheless, similarly to how the "chemistry" relation stat is being currently used.

Tiers, like relation states, are not defined anywhere, and are instead calculated through checks. In the same way that each facet has some minimum thresholds to be considered true, each tier requires the multiples of said basis to compute as complete. To exemplify it, if the facet "friend" requires trust 80, respect 80 and yearning 50, a tier 2 of friendship will only be computed with values of 160, 160 and 100 respectively. If any multiple does not meet the mark, the returned tier is the lower one where all conditions are fulfilled.

All those changes were a compromise between representation of human interaction and flexible, easy-to-use structures, and they are susceptible to change if it turns out to not be an accurate enough model.

7.1.4. World simulator

As it was explained previously, world is constituted by four types of elements: scenes, themes, characters and objects. Each scene contains a vector of the other three types, and the enumeration variable they use to identify themselves is called SCENE_ID. Within this enum, for each type, we have a FIRST_X and LAST_X declaration, a very useful resource if we want to check whether something is of a specific kind (to know which vector to iterate, for example). Besides this useful implementation, everything else was kept as proposed.

7.1.5. Event manager

The only difficulty that was faced by the event manager was to save the union variable appropriately. In its current state, there was no need for it, but its inevitable implementation will require a transcription type for each kind of variable the union can hold. This could be problematic when saving large chunk of data (like an entire scene), so as a preventive measure, it is advised to only use the minimal information (in the case of a scene, send only the scene ID, and retrieve the object with a pointer or by value if necessary).

7.1.6. Theme control

After doing many revisions to the original plan for the theme control, and with a more defined basis for the character structure, many drastic changes were developed. To begin with, the system was given focus under the restraints of the project: since we would not be able to use parametrization to control character progression, we were limited to character narrative being a series of nodes, activated or not based on character action, and independent from each other for the most part (to avoid branched structures; our focus was in player agency). This contrasted with the fear of repetition that guided much of the original draft, despite it being incorrectly considered as a factor that went against depth or meaningfulness.

Retaining the spirit of system independence, characters are still the ones who control how they respond, and how they can evolve in the span of a playthrough. Consequently, it did make no sense that the Theme Control system dictated its vertical or horizontal evolution in the theme map that was originally proposed.

In the same way that the Narrative Structure only decides when things happen, Theme Control only decides which option from all the available ones should be prioritized for the sake of a better cohesion between the actions of the player, and the open branches of the character. This means that characters only consult theme control when there is as a need to answer the player's (or occasionally, an NPC's) thematic actions; any input that does not regard themes should be responded directly by the character's default routines, which are presupposed to be designed with thematic intent.

Those assumptions had a big impact for the two functions of Theme Control:

- Theme Definition: instead of sending information based on the current diversity of themes visited, theme definition would only focus on keeping track of the player's later explored themes. Based on that, output would be given to agents to guide them, ensuring to follow behind player intention or interests. Weights should still dynamically update, but with these priorities in mind.

- Theme Exploration: the premise of this sub-system stemmed from the desire to provide new content throughout time. While novelty remained important, once it had been relegated to a second place, this sub-system seemed to lose any relevance. After all, if the character should only be accountable for player action consistency, and it is already accounting for narrative consistency within itself, what relevance would it have that the

player was in the proper state to interact with a lower tier thematic event? If the character run out of adequate nodes, Theme Definition would give all its continuations a low priority, making it difficult for the character to enter the narrative cap of active storylines, and their narrative-relevant events would never be progressed, at least not until player action changed that. This functioning was judged perfectly fine, and in fact, the necessary one, and as such, Theme Exploration was considered unnecessary, and consequently deleted from the system.

To adapt to the new functions of the module, we designed a new set of data structures: an enum of themes, and a struct of event thematic value. With these two pieces, we would arrange a multimap, with the first key being the theme enumeration, the second one being the struct.

Within the struct, there would be three variables: an event class that defined the character event (for example, helped_master_at_storehouse), with another event class that defined how the player responded (for example, ignored_event), and a weight. The weight would serve to indicate the expected emotionality of the event, based on the response it received.

It is important to note, however, that since there are few player interactions implemented due to time limitations, the Theme Control module is an empty structure. Despite our focus on player agency and thematic relevance, as we will explain in our conclusions, this is not an impediment to use or understand the system.

7.1.7. Narrative structure

The entire sub-system of narrative structure was omitted in the implementation due to multiple factors:

- Its function is to regulate the actors in the world, thus it did not make sense to start implementing it without any actors.

- Its addition could be done a posteriori, so not implementing it would not impact negatively on any other sub-system.

- There was not enough time to develop it without omitting proper research conclusions, which were considered to add more value than the module could provide.

7.2. Yacchatana

Complementing *Yume* will be *Yacchatana*[63], the name of the "game" that will serve to showcase our systems functionalities. Considering the time restrictions and the purpose of the research, *Yacchatana* has almost no playability at all, and is designed on the same principles as *Yume*, those being fast iteration with the minimal effort wasted, as it is not a creation conceptualized as a prototype, but a space for *Yume*'s functioning.

To better understand the capabilities of *Yume*, and how they were exploited, we will proceed to do a run-through of the game, followed by the most interesting details of the final version.

Yacchatana is a series of menus that simulate interaction with a code-simulated world filled with agents, with the user as the main character. To quickly summarize the premise, the player lives stuck in a city hanging on top of a bottomless pit, feebly sustained by the walls that circle the abyss. The main conflict stems from the confrontation between a faction of misfits, and the protagonist's father's guild. For more narrative context, *Yacchatana*'s debug menu has more explanation, although most of the narrative work that was made ended up unused in favor of feature development.

[63] For those adept in Japanese, yes, this is an adaptation of やちゃった, and ample liberties were taken to transcribe it into romaji, it's not one-to-one.

Figure 19 Yacchatana *world map sketch*

In terms of mechanics, the game allows the user to Attack, Provoke, Move To, Talk To and Look Around. All said actions are presented as commands, but would be envisioned in-game as a string of sub-mechanics (or lack of) being performed; for example, “Provoke” could be signaled through an animation (similarly to *Dark Souls* gestures), or with a text prompt, it’s not necessarily a specific controller input.

7.2.1. Demo menus

Compiling *Yacchatana.exe* opens a *Yume* window that provides information about the monitored game's narrative, divided in two blocks: "Debug" and "Yume debug window".

The first is content that is accessed outside of *Yume*, simulating a game's functioning:

- Debug options: values that define how game debug info is processed and stored, although currently the only important features that are functional are the game saving and loading.

- Game events: the most important tab, where users can simulate player action to affect the state of the world. They are divided into "Generic player actions", which offers the possibility to use basic mechanics on the characters that are found on the currently selected scene (which can be changed within this same tab); "Scene actions", which are those limited to the current scene (the only one available is in the "Home" scene); and "World actions", those that simulate world events external to player control (only "Earthquake" should be available).

- Game world info: a short guide on the narrative context of *Yacchatana*. It includes some notions on the demo's world-building, but is non-relevant to the use of the tool.

The second window refers to content internal to *Yume*, thus, that represents the structures used to manage narrative context in real-time:

- Scene elements: a list of all the scenes, with the character, objects and themes registered in them.

- Character info: a list of all relevant character data. Since characters are quite complex entities, their information is divided into multiple segments:

    - Past events: whenever a relevant event has to be stored, it gets placed here. Currently not being used.

    - Personality: describes the values, from -1 to 1, that define a character's personality. Can be manually altered by the user.

- Relationships: a list of how the character relates to others, defining all the components for each connection. All said values are editable in real-time; if the right combination of values are introduced, the type of relationship showcased under the character's name will adequately update.

- Behavior: defines the internal action states of the character, showing its goal, how it is progressing, and the actions left to reach completion. Since a lot of the applications' events are timed, a button "Open all behavior flags" was added atop the "Character info" tab to facilitate accessing that information quickly.

7.2.2. Event list

The following will be a summary of the different types of events implemented, as well as a list of all of those that can be currently triggered from the *Yacchatana* demo. Since they were added to display different available features, their narrative context will be ignored in favor of what they can add to the experience a developer might want to craft. All the mentioned events have some kind of debug feedback that can be seen in the corresponding debugging menus, and which will be specified to define their expected behavior.

First we introduce a typology of possible events we support:

- One-time / cyclical events

- Reactive / timed events

- Action-trigger / stat-modifier events

- Player-reactive / character-reactive events: while they could appear the same at surface-level, they are internally very different, since player-reactive events come from *Yacchatana* at uncertain moments, while the character-reactive ones start and end in *Yume*, and can be regulated by the narrative structure module.

- Goal-chained events: events that trigger a series of goals in succession, but which are defined separately, as they might have to be called individually under other contexts.

- Multi-character events: events that affect multiple characters; not to confuse with joint actions, which are not yet available.

The described diversity of events can be seen in the different narrative beats implemented (unless it is specified otherwise, it can be assumed that feedback from the event can be seen at "Yume debug window → Character Info → (character name) → Behavior" menus):

- ANGEL_APPEARS: after fifteen seconds from the start of the application, "Angel" will be moved from scene "Limbo" to scene "Home" (observable in "Yume debug window → Scene Elements" and "Debug → Game events" if player is in "Home" scene).

- INVESTIGATE_A_WAY_OUT: every 20 seconds, "Master" will be occupied with its research, until a new goal or reaction is sent in.

- ATTACK, PROVOKE and TALK: from "Debug → Game events → Attack / Provoke / Talk", the user can choose to use one of such actions on one of the current scene's characters. Doing so will reduce or increase their relation stats in different amounts, depending on the internal parameters between the protagonist and the character.

- LOOK_AROUND: if the button "Look around" (under the Attack – Provoke – Talk options) is pressed while in "Father Guild" scene, the protagonist will find certain object that will cause "Kilva" to turn against its boss. To see this reflected, one can see "Kilva"'s behavior data, where its goal will turn into "Sabotage the cave faction". Such change will happen regardless of current goal, since it is a reactive goal (thus, that causes an immediate interruption) with a GUARANTEED_PRIORITY value. Once the goal is completed, "Kilva" will be moved to the "Limbo" scene, which can be seen from "Yume debug window → Scene Elements". Additionally, the character will "freeze", meaning it will not respond to any events it would usually receive goals from (like the next event).

- WATER_STEALING_PLOT_REVEALED: by pressing the button "Discover water stealing" on "Debug → Game events → Scene actions" (current scene has to be "Home", as it is an action that should only be performable there), two event branches

will activate. Firstly, "Father" will enter the REACH_AGREEMENT goal, which, after thirty seconds will cause the character to loop through goal REPEL_KILVA_ATTACKS. Secondly, "Kilva" will start OBTAIN_EXPLOSIVES goal whenever its previous goal is completed (successfully or not).

- EARTHQUAKE: the one option in "Debug → Game events → World actions", it causes all objects to be destroyed. To see this reflected, "Yume debug window → Scene elements → Father Guild", where there should be two placeholder objects before pressing the button.

- KILVA_GET_SUPPLIES & TERRORIZE_TOWN: "Kilva" will, unless the sabotage event is triggered, alternate between those two goals. TERRORIZE_TOWN has a 90% chance of "Kilva" attacking the "Master Workshop" scene, which will trigger the WORKSHOP_BURNED event. In response to that, "Master" will immediately (again, using a reactive goal) enter RECOVER_FROM_ARSON, and will stay in that state until he reaches a "Friend" relation type with "Farmer" character. To facilitate this happening, one can access "Yume debug window → Character info → Master → Relationships → Relation with Farmer", and change the value of "Trust, "Respect" and "Yearning" to 100 (a "Friend" text should appear atop the relation sliders). If such condition is fulfilled, "Master" will be taken in by "Farmer", and will finally be able to resume its investigation.

7.2.3. Scene management

We do not have fully implemented save functionality for the scenes; being a showcase of *Yume*'s capabilities, we do not expect the player to actually "play" the game. However, because scene loading will be used regardless of mostly any context, we made a system to load a list of scenes that can be manually updated from a .txt, following the structures implemented, where one can choose which scene to begin from, and what elements are in the scene.

In terms of how the game correlates entities with *Yume*, we decided to use *Yume*'s own SCENE_ID enumeration, making a header for data that had to be shared between both parts of the application, so that it could be included without adding dependencies. One caveat this method has for designers, however, is that if they take an object from the notepad files, give it

ID = 23, and then they add an enum member prior to that, new ID will actually become 24, so the game will load an incorrect entity. The alternative was to have the code work with strings, but we settled for the enumerations, since they are faster to work with. A decent compromise would be to have a string – ID conversion function, but that would imply updating certain structures every time an agent (object, character or theme) was added, which was too inconvenient at large scales to make a mandatory implementation of it.

8. Conclusions

8.1. On the project

To make a quick summary, *Yume* was a relative success, while *Yacchatana* was a bit of a disappointment. Since they were to distinct parts with different objectives, we will discuss them separately.

8.1.1. Yume

Yume was the part of the project that had the most time spent on, as it was the base of the entire research. While the first three months of research went more or less as the planning envisioned, the design phase took a lot more time than it was inferred, partially due to the surprisingly taxing workload of other subjects from the degree. The worse effects of external work were not so much the time it required, but how disruptive they were, forcing us to spend one or two days just to reconnect with the project's open threads, before we could get back to working at full force and consciousness of what was being done.

Making matters worse, the complexity of the system forced us to rework many of its features, some having to be redefined multiple times, which was obviously costly in time, but more importantly, in motivation. Worrying that something that did not have a practical basis to begin with was not going to work, and which forced to alter something that was previously established as an adequate implementation, bred hesitation and doubts on the project's viability, which in turn affected how we foresee the rest of the development.

This sense of stagnation also bolstered laziness, making important aspects seem less necessary to define at the moment, in exchange for faster progress. Fortunately, the experience we gathered in many of our previous projects served to walk the fine line between designing the elements that were necessary pre-production and those that had to be developed in-production, and overall, we judged a fine performance in this ambit.

What was not so gracefully managed was our motivation in terms of what the project evolved into, which will be discussed later, 8.3. On thematic integration. As a result of both these successes and failures, *Yume*' first iterations were completed successfully, while not reaching the final stages that would make us consider it a finished product.

To judge the success of the tool, we established a series of goals at the beginning of the research, which we can now recover to examine the results (we will add an indicative score from 0-10 to represent how the research adapted to the expectations we had of each section):

- Input flexibility: the usage of events and unions proved successful in this regard, although they also carried some downfalls that will be discussed later on: 9 / 10.

- Input complexity: adding different types of variables to the event union facilitates input complexity, but a posteriori we thought of other structures that could improve usability from its current (in this specific aspect) unfortunate state, to an acceptable, and even remarkable one: 6 / 10.

- Reaction persistence: this has been both a hit and miss, because, superficially, we have the means to enable it, but the moment in which the state persistence has to be read or distributed across systems, it becomes a more complicated matter to operate: 6 / 10.

- Reaction interconnectivity: while the narrative state was something that became diluted with the simulative aspects of the application, the communication that characters offer, and how easily they can be retrieved, facilitates narrative interconnectivity: 9 / 10.

- Ego layers: despite factions not being implemented[64], the personality – relationships – planning triad that manages character state is incredibly effective at separating the different facets of a character, albeit it could use better usability: 8 / 10.

- Output flexibility & complexity: the usage of the event manager greatly solves this issue, allowing for any kind of output from a limited amount of input: 10 / 10.

- User accessibility: the fulfilled ethos of system independence greatly paid off for accessibility, making self-contained functionality that can be understood one at a time, piece by piece. Additionally, clean coding that ensures that each piece can work as

[64] Since they can be understood as a character within a character, their future implementation was not considered a far stretch from what we have currently.

independently as possible makes it so that users can master the system at their own pace, instead of forcing them to understand its whole to begin working with it: 9 / 10.

The most important condition to judge, however, is not how effectively the system adjusts to what were our expectations, but whether it is capable of fulfilling the role it was created for. In summary, how realistic it is to implement as a dynamic narrative controller in a real project. As with mostly every other tool, the answer depends on the project, so here we present a series of guidelines based on the analysis of *Yume*'s strengths and flaws.

As a preemptive disclaimer, the assumption that we are presenting contexts in which *Yume* could be used implies that, indeed, it can be used as a dynamic narrative controller. Even then, there are many flaws that we think are not inherent to the system's design, and many features yet to implement, and as such, we would not recommend an immediate addition to a project's toolkit, even if the current version is better than the lacking libraries currently available for the task.

A problem that *Yume* could not solve is the matter of authorial burden in narrative design, so dynamic plots with a lot of one-note interactions will make a very smooth use of it, while stories with very intertwined events that are hefty to structurally describe will not. Games with a lot of mechanic variety and interactivity will have an easy portrayal of state and a clear structure to work with, allowing for minimal complexity, but with an authorial burden that could be decreased by taking other project-oriented approaches.

In fact, we consider one of *Yume*'s greatest strengths to be, too, one of its greatest weaknesses: because the system is so specific in its objective, many games would probably do better with using simplified versions of what we developed, limiting the scope of who we can reach. This was already established when discussing how the *Yume* approach would be to give total design freedom, and then add limitations that users could enable to fit their vision, and is an inevitable part of trying to solve an inherently complicated problem like dynamic narrative control.

On the other hand, *Yume* offers the specificity that is hard to achieve in short development cycles, remaining flexible, portable, usable and accessible across most of its modules. To top it off, it has extensive run-time debugging options that are completely internal to *Yume*, thus speeding up the development process regardless of where it is implemented.

In summary, *Yume* is an unpolished tool that has potential to go from decently useful to fantastic, a direct translation of the project's research and design phases mixed with the difficulties that emerged along the way.

8.1.2. Yacchatana

We described *Yacchatana* as a bit of a disappointment, an opinion that stems from two places: firstly, the unreasonable expectations that had been placed on our part into it. Secondly, and more importantly, it was a very flawed concept from the beginning, and in a way, it had to be.

Yacchatana was envisioned as a fully-playable narrative, with multi-event plotlines incorporated, multiple endings, a plethora of characters, its own world, mechanics with which to interact… And in fact, many of those elements were developed alongside *Yume*. Unfortunately, we did not make it in time to make enough progress that it would make sense to add those elements into this report, so most of them were eventually removed. In the end, it was reduced to a playground of event trigger – event response, without much cohesion.

A factor that contributed to it was that the need for creative expression, which emerged during the research's process, advancing the development of the "game" into generating structures (like an entire thematic tree) that would later on become unusable due to some of *Yume*'s redesigns.

On top of that, *Yacchatana* is not a real game: it only simulates being one, so in many occasions we had to code behavior that was not intuitive to what we needed, in order to adapt to a game expected behavior (for example, we needed a notepad file that held scenes, so we could load them at run-time, to simulate scene asset traversal at the start of a play session).

Approaching its development with those limitations, however, was a requirement to *Yume*'s success, because there would be no time to develop a real game's environment, and even less to ensure it functioned properly if non-narrative systems began taking time to be developed.

All in all, *Yacchatana* fulfills its purpose, but leaves an unremarkable impression in terms of presenting the achieved results.

8.2. On implementation

With the high-level conclusions out of the way, we can bring attention to an unexpected practical problem that was realized after the project's conclusion.

From a design standpoint, we intended to follow very structural formulas: divide everything in small pieces so users can wrap their heads around each piece of functionality. We cannot judge the willingness to keep system independence as a mistake, yet that approached ended up affecting certain modules negatively.

By failing to identify the importance of character action after the theme's downfall (which will be discussed in the next section), we ended up with a very restrictive action-state system, which would be the only module in need of being reworked. It simply does not work on the user's favor, and that is unacceptable: one same action can have different contexts, yet in trying to separate action from the context, we end up forced to hard-code data that increases authorial burden beyond what should be acceptable.

In fact, for this kind of system, it was observed that the community's discussion around context tracking is not proportional to the importance it holds, for consistency and progression alike. This was also clear to see in event storage, which could end up holding relatively big chunks of data because very specific needs had to be fulfilled to properly carry on very specific events. Then this data had to be replicated in multiple places so that each system could, independently, have "perfect" knowledge of the simulation's state.

A better approach would have been to make a system that can centralize information, while keeping independent systems so they can perform their tasks without creating dependencies. In summary, have a centralized system with two brains, instead of fully relying on events: one being the Manager Manager to control the functioning of the system, the other a Context Dictionary module or so, that can be easily accessed by everyone to gather information about the diegesis.

8.3. On thematic integration

To end the conclusions, we will proceed to address the elephant in the room, and the most important part in regards to the focus of this research: how come a dynamic narrative system based on themes has a theme control module that is of barely any use to the narrative

progression. The answer is quite simple: themes are found in the rest of elements of the system, impregnating the diegesis instead of being a constricted part of it, a natural result of the essence of themes.

Such approach, we considered insufficient, however, because themes were expected to be a web of interactions set by the designers to bypass authorial burden, while keeping authorial intent, so that the players could hold a conversation with the system. That was the appeal of *Yume*, even if it can be given other narrative uses. Even then, the causes that would impede such development from being reached where clear before *Yume*'s first line of code.

The theme's downfall, as we referred to it previously, occurred at the beginning of the design phase. The original idea was to have a parametrized set of values. Each action, depending on its context, would alter said values, and the different combinations of parameters would, ideally, reveal information about the player, that we can then use to respond with narrative poignancy.

At the time of truth, however, we found no feasible way to apply the idea to code, because the decomposition of a theme into quantifiable metrics is so complex and varied on a case by case basis, and more importantly, to do so meaningfully (or in this case, not simplistically), that a proper implementation would require the user to build everything for their specific idea by themselves. And if that was the case, we could only provide generic structures to control theme progression manually, which is the option we resorted to through events.

Regardless of how this crucial change ended up affecting the rest of the system's design, causing the first big re-work on many levels, we wanted to focus on how useful the project has been on better understanding the possibilities of a theme-based implementation. Not giving up on the idea of a videogame tool that facilitated the conjunction between play and narrative themes, and with the experience gathered by making *Yume*, the conclusions seemed the proper place to return to the original question, and revise any other option that could have made sense.

Abstractly, the pipeline that governs our system's functioning is the following: there is a cumulus of states, the variation of which sends signals that alter the behaviors of the state-regulating systems. Said signals can serve to enable new behaviors, or to disable them. Player interaction is based on the feedback that said behaviors give: if a signal does not translate to a change in the world, it can never become part of the "game", that is, the cumulus of states perceived in the player's mind. As a natural result, then, we based system interaction around the enabling, disabling and prioritization of behaviors.

It was mentioned in the theme typology that there is a kind of theme exemplified by character actions; while this remains true, certain assumptions made in this regard led us to fatal conclusions. The train of thought was such as "if a game revolves around actions", taken here as a close parallel to behaviors, "and actions have potential for theme depth, then a system that regulates actions, oriented towards a thematic facet is possible".

What was realized during the development of *Yume*, however, was that it is not actions that themes are tied to, but to the contexts where those are performed; not that themes are affected by the immediate narrative context (the way they are presented, although they are), but that the coupling of a theme with an action is not enabled by the action itself, but by the context in which it takes place. This was made very clear when programming the HTN planning system, as one action could trigger multiple events of very distinct theme fields.

To logically justify this disconnect, we must remember that themes are subtextual entities: they are a construct of the reader's mind, similar to the notion of art, which can be more or less guided by the narrative elements. To put it bluntly, ideas, sensations or feelings will never cause observable change until action is taken, yet the actions taken to fulfill them are not necessarily related in any aspect that is not contextual.

A simplified example would be the following: we are making a story about parental love. If the narrative handler receives the action "killedEntity", the system should adequately respond by updating the cumulus of states, both in world and character. When having to update the themes, though, inferring whether an entity has been killed to protect an offspring, in self-defense, to gain something, for fun, etc. becomes impossible without a context definition. And by this point, whatever contexts will be implemented is a purely specific matter, outside our generic scope (although even for particular projects, it probably ends up being more efficient to hard-code a multitude of contexts, than to try to parametrize them).

To our knowledge, there is only one way to correlate behaviors and themes, which is to define the behaviors of the game mechanics as thematic abstractions: in performative contexts, like it is the case for *Fire Emblem: Thracia 776* or *Dark Souls I*, where the theme is the struggle to succeed in specific fields, the player matches the theme, because they are experiencing it through the act of play. In those cases, however, there is no need for theme control, because the entire game is managing themes through its own rules of mechanics and dynamics.

In summary, there is no instance in which dynamically tracking themes is possible at the level of complexity that an adult-catered story should.

8.4. On authorial burden

The impossibility of significantly tracking theme progression directly implies that we cannot make a system capable of self-generating a story based on the narrativity of player input[65], at least not without a predefined pool of manually authored content that covers enough space of possibilities as to induce a feeling of agency.

In turn, authored content implies authorial burden, a limitation that *Yume* prioritized as an objective to surpass. Throughout the project's implementation, there have been many examples of techniques that can reduce such burden, and if we cannot provide a solution native to the system's design, the least we can do is to indicate different uses *Yume* could offer to improve narrative progression's scope:

- 1-to-1 event response: the basic narrative progression that has been implemented, consisting on triggering an event when a set of conditions is fulfilled. Using this kind of events does not improve authorial burden in any way.

- Accumulative triggers: one or a series of events augment the value of a variable until it reaches its cap point, triggering the event it holds. It would have a similar functioning to character relationship definition, but without the tier system, and oriented to narrative progression. A tool best for important events, where a sense of build-up and effort is more rewarding than a causal relation.

- Probabilistic triggers: adding a random value to the execution of an event response, useful both to simulate events that do not always happen, and to add breathing space between narrative beats. It would become especially useful if it could be influenced by narrative structure module, where it would probably be held.

- Mutually exclusive content: cluster of events that are mutually exclusive, so that developers can have a point of reference to what possible states the game could be in

[65] Due to narrative's lose definition, we could make a "story" based on responding to player performance on a purely mechanical level, but that would be more alike dynamic challenge generation than the emergent narrative we want to achieve.

after a certain point. A very powerful tool to avoid an exponential increase of narrative possibilities.

- Polling: triggering pre-defined events, at the proper time, the result of which depends on world state instead of player actions (although the world state is defined by player action). A way to have players "deal" with their actions, defining limited possible outcomes, re-contextualizing the narrative options the game can offer (which in turn affects how it can be progressed), both fitting more traditional narrative structures of rise to climax, and reducing authorial burden.

8.5. Closing statement

The research surrounding *Yume* has been very fruitful in the definition of which paths hold potential for the future of interactive narrative, and which do not. While the design phase provided a first, well-guided impression, the project's development handled solid evidence, both theoretical and practical, of the impossibility of a theme-based dynamic narrative controller system.

On top of that, it has provided a practical system that, despite its flaws, has been a solid investment of time and effort for future projects that can take advantage of the untapped potential that interactive narrative holds, by facilitating access to a usable, effective and quick-to-implement tool.

9. Future work

From our conclusions it is clear that there is many work left to do; the following section will be, therefore, a brief discussion of those possibilities.

9.1. Yume *improvements*

Yume still needs a bit more work to be a fully usable narrative tool, and also has some improvements that would make it excel at what it does:

- Missing implementation: factions, character grudges, narrative regulation, world connectivity, joint actions, etc. In addition to the aforementioned Context Dictionary, which could also be added for progression consistency and information retrieval, although the design effects it could entail would have to be considered first.

- Improvements of existing features: if a Context Dictionary was implemented, (mainly) events would need a rework, changing the union variable for some other kind of data holder that redirects contextual clues to the Dictionary.

  Another minor improvement would be to have better enumeration structures, since we currently have too many enumeration types that sometimes overlap each other, when they should be hierarchical, from general to specific uses.

  Finally, we could try to have World Simulator's object, character and themes vectors outside of scenes. The reason for this is that we noticed that, many times, we want to access where an element is, instead of checking all elements of a scene. Having the SCENE_ID enumeration organized to recognize the type of an element based on its ID, our current approach might have room for optimization (we could still have scenes in a vector themselves, and have a function that returns all elements of a scene, even if it is occasionally slightly slower).

- Necessary changes: as it was previously explained, actions should be self-regulated, or at least keep some kind of reference to a structure that provides the context in which they have to be executed.

9.2. Yacchatana *improvements*

On this regard, the following steps are clear: gather all the content that could not make the cut for the first iteration, and take the time to implement it, diversifying the nodes the user can explore and growing the base to personalize the experience. Any addition done in this aspect, however, would be to improve the experience of *Yacchatana*, since the showcase of functionalities, which was its initial purpose, is practically completed, with several examples in code.

10. References

Abott, H. P. (2008). *The Cambridge Introduction to Narrative.* doi:https://doi.org/10.1017/CBO9780511816932

Alphonso, N. (2009). *Killzone 2.* Guerrilla Games.

Araki, H. (1987-current). *JoJo Kimyou na Bouken.*

Ashwell, S. K. (26 de 01 de 2015). *These Heterogeneous Tasks*. Obtenido de https://heterogenoustasks.wordpress.com/2015/01/26/standard-patterns-in-choice-based-games/

Bosch, H. (1515). *The Garden of Earthly Delights.*

Capcom. (2001-2015). *Ace Attorney.*

Cardoso, P., & Carvalhais, M. (2013). Breaking the Game: the Traversal of the Emergent Narrative in Video Games. *Journal of Science and Technology of the Arts*. Obtenido de https://www.academia.edu/16375882/Breaking_the_Game_The_Traversal_of_the_Emergent_Narrative_in_Video_Games

DidYouKnowGaming? (2019, 11 16). *Satoshi Tajiri: How Pokemon Was Made - Did You Know Gaming Ft. Furst*. [Video file]. Retrieved from https://www.youtube.com/watch?v=G5aBg6GFufI

Dubbelman, T. (2016). Narrative Game Mechanics. *Interactive Storytelling: 9th International Conference on Interactive Digital Storytelling.*

Dwarf Fortress Wiki. (05 de 03 de 2022). Obtenido de https://dwarffortresswiki.org/index.php/DF2014:World_generation

ElectronicArts. (2000-2021). *The Sims.*

Evans, R. (24 de 04 de 2002). *GDC 2002: Social Activities: Implementing Wittgenstein*. Obtenido de https://www.gamedeveloper.com/design/gdc-2002-social-activities-implementing-wittgenstein

FailbetterGames. (17 de 07 de 2012). *Patterns of Choice*. Obtenido de http://wiki.failbettergames.com/patterns-of-choice

FailbetterGames. (s.f.). *StoryNexus: Reference Guide*. Obtenido de https://docs.google.com/document/d/1K1wnNJoBhxr17fe3kHQTnpvWLdyxeKWZBKivDQHsdJg/edit#heading=h.q4vgzctmcjto

Fallout: New Vegas Companions. (s.f.). Recuperado el 07 de 10 de 2021, de https://fallout.fandom.com/wiki/Fallout:_New_Vegas_companions

Fallout: New Vegas Karma. (s.f.). Recuperado el 07 de 10 de 2021, de https://fallout.fandom.com/wiki/Karma_(Fallout:_New_Vegas)

Fallout: New Vegas quests. (n.d.). Retrieved 10 7, 2021, from https://fallout.fandom.com/wiki/Fallout:_New_Vegas_quests

Fallout: New Vegas Reputation. (s.f.). Recuperado el 07 de 10 de 2021, de https://fallout.fandom.com/wiki/Fallout:_New_Vegas_reputations

FromSoftware. (2015). *Dark Souls II: Scholar of the First Sin.*

FromSoftware. (2016). *Dark Souls III.*

FromSoftware. (2018). *Dark Souls: Remastered.*

GameFreak. (1998-2021). Pokémon.

GameMaker'sToolkit. (2021, 01 28). *How the Nemesis System Creates Stories*. [Video file]. Retrieved from https://www.youtube.com/watch?v=Lm_AzK27mZY

Garbe, J., Kreminski, M., Samuel, B., Wardrip-Fruin, N., & Mateas, M. (2019). *StoryAssembler: an Engine for Generating Dynamic Choice-Driven Narratives.* Obtenido de https://mkremins.github.io/publications/StoryAssembler.pdf

Gaut, B. (2000). "Art" as a Cluster Concept. En N. Carroll, *Theories of Art Today* (págs. 25-44). Madison: University of Wisconsin Press.

Gervás, P. (07 de 07 de 2009). Computational Approaches to Storytelling and Creativity. *AI Magazine, 30*(3). doi:https://doi.org/10.1609/aimag.v30i3.2250

Gervás, P., Concepción, E., León, C., Méndez, G., & Delatorre, P. (2019, 01 30). The Long Path to Narrative Generation. *Journal of Research and Development*, 9. doi:10.1147/JRD.2019.2896157

GreatArtExplained. (16 de 04 de 2021). *Hieronymus Bosch, The Garden of Earthly Delights (Full Length): Great Art Explained*. Obtenido de https://www.youtube.com/watch?v=vBG621XEegk

HALLaboratory. (1999). *Pokémon Snap.* Nintendo.

Hammond, S., Pain, H., & Smith, T. J. (2007). *Player agency in interactive narrative: audience, actor & author*. Newcastle, UK.

Harrell, D. F., & Zhu, J. (2009). *Agency Play: Dimensions of Agency for Interactive Narrative Design.* Obtenido de https://www.researchgate.net/publication/221250432_Agency_Play_Dimensions_of_Agency_for_Interactive_Narrative_Design

Hoefer, C. (21 de 01 de 2016). *Causal Determinism*. Obtenido de https://plato.stanford.edu/entries/determinism-causal/

Houlgate, S. (20 de 01 de 2009). *Hegel's Aesthetics*. Recuperado el 27 de 10 de 2021, de Standford Encyclopedia of Philosophy: https://plato.stanford.edu/entries/hegel-aesthetics/#Con

Howard, T., Pagliarulo, E., Nesmith, B., & Kuhlmann, K. (2011). *The Elder Scrcolls V: Skyrim.*

Hubbard, C. (2005). *FEAR.* Monolith Productions; Wargaming Chicago-Baltimore.

Hunicke, R., LeBlanc, M., & Zubek, R. (s.f.). *MDA: A Formal Approach to Game Design and Game Research.* Obtenido de https://users.cs.northwestern.edu/~hunicke/MDA.pdf

Inform7 Documentation. (s.f.). Obtenido de http://inform7.com/book/WI_1_1.html

IntelligentSystems. (1990-2019). *Fire Emblem.*

Kaga, S. (1999). *Fire Emblem: Thracia 776.* Intelligent Systems.

Koenitz, H. (25 de 02 de 2018). *Narrative in Video Games.* Obtenido de https://www.researchgate.net/profile/Hartmut-Koenitz/publication/322893685_Narrative_in_Video_Games/links/5a929bf10f7e9ba4296e5ade/Narrative-in-Video-Games.pdf

Koenitz, H., Ferri, G., Haahr, M., Sezen, D., & Sezen, T. I. (2017). *Interactive Digital Narrative. History, Theory and Practice.* Routledge.

Kreminski, M. (2020). *Designing to Support Authorship Play in Emergent Narrative Games.* Obtenido de https://escholarship.org/uc/item/47n0221v

Legault, L. (2017). Self-Determination Theory. En V. Zeigler-Hill, & T. Shackelford (Edits.), *Encyclopedia of Personality and Individual Differences.* doi:10.1007/978-3-319-28099-8_1162-1

Levine, K. (2014). *Ken Levine On 'Narrative Lego'*. [Video file]. Retrieved from https://www.youtube.com/watch?v=p40p0AVUH70&t=2459s

Lima, E. S., Feijó, B., & Furtado, A. L. (2014). *Hierarchical Generation of Dynamic and Nondeterministic Quests in Games.* Pontifical Catholic University of Rio de Janeiro, Department of Informatics. doi:10.1145/2663806.2663833

Lively, G. (2017). Anticipation and Narratology. En *Handbook of Anticipation.* doi:https://doi.org/10.1007/978-3-319-31737-3_7-1

Lucat, B., & Haar, M. (2015). What Makes a Successful Emergent Narrative: The Case of Crusader Kings II. *International Conference on Interactive Digital Storytelling*, (pág. 8). doi:10.1007/978-3-319-27036-4_25

Mambrol, N. (20 de 02 de 2021). *Analysis of Dante's Divine Comedy*. Obtenido de https://literariness.org/2021/02/20/analysis-of-dantes-divine-comedy/

Mason, S., Stagg, C., Wardrip-Fruin, N., & Mateas, M. (2019). Lume: A System for Procedural Story Generation. San Luis Obispo, CA, USA.

Mateas, M. (2001). *A preliminary poetics for interactive drama and games.* Obtenido de https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.576.5683&rep=rep1&type=pdf

Mateas, M., & Stern, A. (2002). A Behavior Language for Story-based Believable Agents. *IEEE Intelligent Systems, 17*(4).

Mateas, M., & Stern, A. (2003). Integrating Plot, Character and Natural Language Processing in the Interactive Drama Façade. *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE '03).* Darmstadt, Germany.

Mateas, M., & Stern, A. (2004). Natural Language Understanding in Façade: Surface-text Processing. In S. Göbel, *Technologies for Interactive Digital Storytelling and*

Entertainment. Heidelberg, Berlin. Retrieved from https://doi.org/10.1007/978-3-540-27797-2_2

Mateas, M., & Stern, A. (2005). Structuring Content in the Façade Interactive Drama Architecture. *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2005), 3.* Marina del Rey, California.

McKee, R. (1999). *Story: Style, Structure, Substance, and the Principles of Screenwriting.*

McRae, E. (s.f.). *Procedural Narrative: The Future of Video Games*. Obtenido de https://www.edmcrae.com/article/procedural-narrative

Merrin, J. (2013). *Exploring Dynamic Interactive Narrative.*

Murray, J. H. (1997). *Hamlet on the Holodeck: the Future of Narrative in Cyberspace.* New York.

Nau, D. (2016). Hierarchical Task Network Planning. En M. Ghallab, D. Nau, & P. Traverso, *Automated Planning and Acting.* Obtenido de http://www.dia.fi.upm.es/~ocorcho/Asignaturas/ModelosRazonamiento/PresentacionesClases/planning07.pdf

Nau, D. (2016). HTN Planning. En M. Ghallab, D. Nau, & P. Traverso, *Automated Planning and Acting.* Obtenido de http://www.cs.umd.edu/users/nau/apa/slides/htn-planning.pdf

Orkin, J. (2006). *Three States and a Plan: The A.I. of F.E.A.R.* Obtenido de https://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf

Outer Wilds (PC Version) [Video Game]. (2019). Mobius Digital.

Payne, M. T. (2014). *War Bytes: The Critique of Militainment in Spec Ops: The Line.* doi:https://doi.org/10.1080/15295036.2014.881518

PeopleMakeGames. (19 de 11 de 2020). The System Behind Hades' Astounding Dialogue. Obtenido de https://www.youtube.com/watch?v=bwdYL0KFA_U&t=653s

Propp, V. (1968). *Morphology of the Folktale.*

Rouse III, R. (2016). *Dynamic Stories for Dynamic Games: Six Ways to Give Each Player a Unique Narrative*. [Video file]. Retrieved from https://www.youtube.com/watch?v=SsSh62mSPZE

Rouse III, R. (28 de 4 de 2016). The Church in the Darkness - Richard Rouse III Interview. Obtenido de https://www.youtube.com/watch?v=nkPZfbY5V0k

Ryan, J., Mateas, M., & Wardrip-Fruin, N. (2015). Open Design Challenges for Interactive Emergent Narratives. En *Lecture Notes in Computer Science* (Vol. 9445). doi:10.1007/978-3-319-27036-4_2

Ryan, M.-L. (2006). *Avatars of Story* (Vol. 17). University of Minnesota Press. Obtenido de https://www.academia.edu/26436067/_Marie_Laure_Ryan_Avatars_Of_Story

Stein, N. (01 de 12 de 1982). The Definition of a Story. *Journal of Pragmatics, 6*, pág. 20. doi:10.1016/0378-2166(82)90022-4

The Elder Scrolls: Skyrim Radiant A.I. (s.f.). Recuperado el 26 de 10 de 2021, de https://elderscrolls.fandom.com/wiki/Radiant_A.I.

TheSaltFactory. (2020, 04 30). *Evaluating Fallout New Vegas companions and side quests- a look at the NCR and the Legion*. [Video file]. Retrieved from https://www.youtube.com/watch?v=JvRUyfNdkdg

Thorson, M., & Berry, N. (2018). *Celeste.* Matt Makes Games, Extremely OK Games, Ltd.

Twine Cookbook. (April de 2021). Obtenido de https://twinery.org/cookbook/index.html

Ventura, D., & Brogan, D. (2003). Digital Storytelling with DINAH: Dynamic, Interactive, Narrative Authoring Heuristic. En *Entertainment Computing* (Vol. 112). doi:https://doi.org/10.1007/978-0-387-35660-0_11

William, M., Harrison, B., Ware, S., Cardona-Rivera, R. E., & Roberts, D. (2012). *Achieving the Illusion of Agency.* Obtenido de https://www.researchgate.net/publication/267225089_Achieving_the_Illusion_of_Agency

Zhu, J., Ingraham, K., & Moshell, J. M. (2011). Back-Leading through Character Status in Interactive Storytelling. Vancouver, Canada. doi:http://dx.doi.org/10.1007/978-3-642-25289-1_4