Narrative Structures and Systems: Leads to Improve the Production of Quest-based Role-playing Games

Antoine Dieulesaint

Abstract
In this thesis I look at the place of narration in video games, through the structures and gameplay systems developed to support narration. I study some examples of each of these and their limitations, focusing on role-playing games. Seeing room to improve quest-based games, I then propose an idea for a quest template to try to solve some of the identified problems.

Résumé
Dans cette thèse professionnelle je m'intéresse à la place de la narration dans les jeux vidéo à travers les structures et systèmes de jeu développés pour supporter cette narration. J'étudie quelques exemple de chaque et leurs limitations, avec une attention particulière pour les jeux de rôle. Faisant le constat qu'il y a possibilité d'améliorer les jeux basés sur un système de quêtes, je propose ensuite une idée de template pour la production de quêtes.

2019

le cnam
enjmin
GOBELINS
L'ÉCOLE DE L'IMAGE

Table of contents

Acknowledgments

I would like to thank Nicolas Esposito for helping me sort out my ideas, Spiders for the opportunity to work at their studio and helping me experience the production of a role-playing video game from the inside, and all the people who have worked on the video games cited in this thesis, especially those who took the time to discuss their work so others could learn from it.

Introduction

Video games have almost always included narrative in their design, knowingly or not. It is a major selling point for many video games, and its structure can become the focus point of its production and marketing. When buying games, players can pick their favorite level of interaction, depending on the agency they want on the story, from a large choice of narrative structures, from linear stories to ones in which the player will change the end of the story by playing the game. Designers have followed suit, by creating systems adapting the narrative of the game without waiting for the player to choose consciously.

In this thesis, “narrative” is understood as a system of interdependent components, and can be broken down this way, according to the distinction first made by Genette [1] regarding literature :
- *story* : the narrative content, i.e. the full set of events as they happen chronologically;
- *plot* : the subset of events presented to the reader, viewer or player, in the order they are presented;
- *narration* : the processes and techniques used to relate a plot to the audience.

In the case of quest-based games, quests are the main way to organise the narrative, as they represent events – or series of events – and contain all the information used by the game system to convey these events to the player, thus creating the narration they will witness. The way the player goes through these quests and in which order will determine the plot of the game.

In this thesis, I will explore some of the narrative structures and systems used in video games, mainly in role-playing games (RPGs). I will then attempt to identify ways to improve the production and integration of narrative and gameplay, and make a proposal for a quest template to answer some of the identified problems.

Narrative structures in video games

A narrative structure is a framework defining how a narrative is presented to the player (in the case of video games) and in which order. It determines how the plot and/or the story of a game evolves after specific player actions. A lot of different narrative structures exist, particularly in video games, a medium fit for narrative storytelling with the players actively taking part in the composition of the plot. Games can also mix and change their structure during the course of their experience.

Linear structures

Narrative structures can be divided into two main categories : *linear* and *non-linear*. The linear ones have in common the unique nature of the presented story : the events of the story and their chronology will stay the same each time the game is played, nothing in the mechanics and structure can change the story and its outcome. This type of structure has been the basis of most media since the beginning of fixed stories, i.e. stories contained in a physical object (a book, a disc, a hard drive, etc.), when it became impossible to alter the story after it reached the readers or viewers. These media in turn inspired video games creators, who applied already existing techniques to their medium.

String-of-pearls model

The “string of pearls” is an early structure typical of video games. It consists of a series of gameplay phases – the pearls – intersected by cutscenes or dialogues to introduce narration – the bits of string between the pearls which tie the whole thing together.

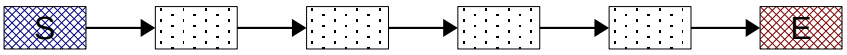


Figure 1: String-of-pearls

S : first scene/event (beginning)

E : last scene/event (ending)

: gameplay phase

: narration (cutscenes, dialogues, etc.)

As one can see, there is only one way from the beginning of the narration to the end. This type of structures has one possible story and one possible plot, and the narration happens linearly. The different « phases » in the game are clearly distinct, and the narration only happens at specific times.

This structure was very popular in earlier games that tended toward telling a story with strong visual elements because it allowed for a better use of the technological and hardware resources available at the time. It is still widely used in games focusing on gameplay (puzzle games, action games, strategy games, etc.), and games who wish to offer players a « cinematic » experience, in which cutscenes and visual effects are the main focus. Separating the moments when the player is in control of the action and moments when the game just tells a story in a non-interactive way allows for the use of gameplay elements which wouldn’t be able to convey story elements as deep as those shown in a cutscene and, conversely, to use visual effects and animations that wouldn’t fit in a gameplay environment.

Recent examples include games such as the *Uncharted* [2] series, in which the player progresses through different levels and events in the story, in a set order. Gameplay itself varies, from platforming to gun fights, but always has the same goal : advance toward the next plot point. It takes some liberties with the structure in a few levels, in which several objectives can be accomplished in any order, but uses systems to circumvent the problems it could cause and make it fit in the string-of-pearls model.

Elastic stories

This term, which started originally as *Bending stories* [3], was coined by Quantic Dream's creative director David Cage, maker of games exclusively focused on stories rather than gameplay mechanics. While he used it in mostly non-linear games, like *Heavy Rain* [4], it is a structure that exists in and of itself. The idea is that the player's choices of interaction dictate which part of the plot will be developed or not.

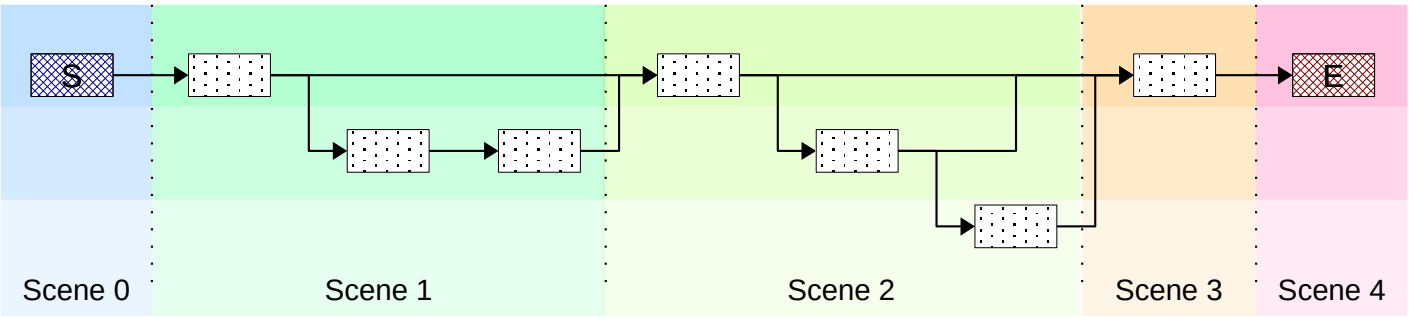


Figure 2: Elastic story

S : first scene/event (beginning) E : last scene/event (ending)

: event

Here, players' actions will determine how much of the story they see or experience. In this example, the shortest path would be to play only three events in the whole plot, while the longest would require players to play seven. Depending on the player's actions in the first event of scene 2, they may or may not play through its second event and, in the same way, may or may not play through its third event.

In this kind of structure, players won't necessarily cause meaningful changes to the story, but they will change the amount of time and level of details allocated to each scene or character, therefore "stretching" the plot like a rubber band, lengthening some parts and shortening others, in turn changing their own experience of the story. This structure is useful to adapt the narrative to the interests of the player without having to ask them what they want to see or not. One way to implement this structure would be, for instance, to offer several mechanics in the same scene, the easiest and most straightforward one being also the shortest: a player with less interest in this particular scene wouldn't try to search for other mechanics and would finish this "less interesting" scene faster. This type of structure also works well when telling several stories concurrently, in parallel.

This structure is present in Telltale Games' games, in particular in their series *The Walking Dead* [5]. In it, it is combined with a string-of-pearls structure : in the gameplay phases, a goal is given to the player, which they can complete straight away, but they can also

wander around, accomplishing optional objectives and witnessing optional plot points, before carrying on with the next scene.

Parallel narration

Telling stories and events that happen at the same time, from the point of view of different protagonists, is an old technique. In video games, it most often comes down to making the plot alternate either between characters, realities, or time periods. The different storylines are usually linked with scenes where the protagonists of several (or all) of them join for a specific event.

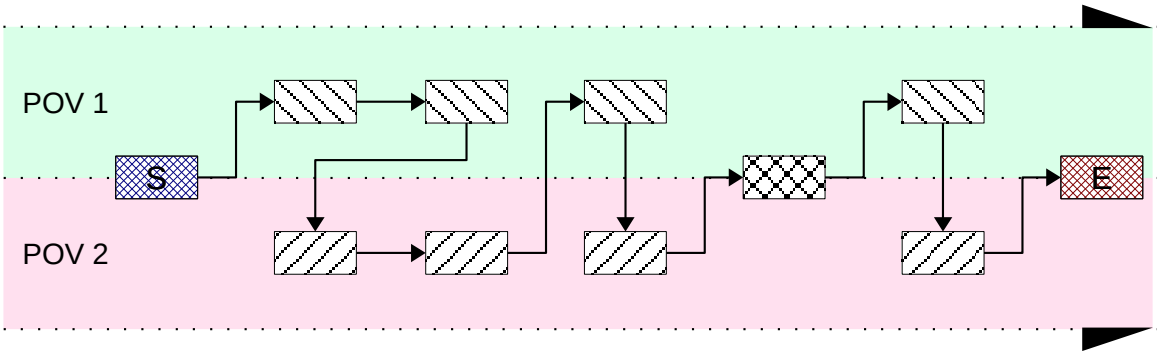


Figure 3: Parallel narration

S : first scene/event (beginning)
E : last scene/event (ending)
: event from POV 1
: event from POV 2

Here, the story is seen through two point of view. The plot starts and ends with scenes in which both points of view are present, with another event in the middle acting in the same way. The plot doesn't follow a chronological order. First the player plays through 2 events from POV 1, then "goes back in time" to play two events from POV 2, and so on.

This kind of structure is used to show several points of view concurrently, and advance the events of the narrative side by side, in order to give a better understanding of the events, or different opinions and versions of those events (as seen in numerous investigation stories and thrillers). It revolves entirely around the plot, and the order in which events are presented.

In *Fahrenheit [6]*, by aforementioned Quantic Dream, the player controls both a murderer and the police officer investigating him, which puts them in a position of knowing much more than any of the individual protagonists, giving them a much better understanding of the events and of the motivations of each character.

These structures are not exclusive to linear narrative at all. While they can suffice on their own as linear structures, they can also be (and usually *are*) mixed with non-linear ones. A string-of-pearls model can branch into several "strings" in the middle of the plot, elastic stories can complete non-linear structures by adding another layer of player agency, and parallel narration even works best when included in non-linear stories, where it makes player choices more interesting as they can be made in one storyline but have consequences on other storylines and characters the player also gets to control during the course of the game.

Non-linear structures

Non-linear narratives have existed for a long time; they originate from oral traditions of storytelling, in which the narrator tells a story and adapts it in real time according to the audience's reactions. Their has been attempts to recreate this kind of narrative in other media, which were met with mixed success. Early contemporary examples include gamebooks, interactive movies like *Kinoautomat* [7], where a live narrator periodically interrupts the film so the audience can choose between two scenes to play next, and of course role-playing games. From the onset of video games existence these games, both in their tabletop and live-action forms, have been one of the main influences, followed by films and novels. Because of the complexity of their rules, role-playing games players and designers soon saw video games as a way to simplify access to them. Video games could emulate the rules without requiring a game master, do calculation impossible in real time with pen and paper, and bring visual interfaces to a medium mostly deprived of it. Thanks to these influences, video games included non-linear narrative structures early on.

Branching stories

The most common technique, directly inherited from gamebooks and role-playing games, found in video games is the branching story. Its main goal being to increase players' agency, this type of structure relies on giving players the opportunity to make choices at several points along the course of the narrative, to influence it. In gamebooks it translates as choices the reader makes at the end of a paragraph, which makes them jump to a new, predetermined paragraph. In tabletop role-playing games it is managed by a set of game rules and by the game master, who reacts to the choices the players make by adapting the story and making changes to the game world. In video games, it can takes different forms depending on which elements of the game compose the nodes of the narrative structure.

These structures can relate to or influence all levels of narrative, sometimes several at the same time :

> *- a branching narrative in the strict sense provides interactive selection of narrated elements conveying particular plot elements in particular ways*
> *- a branching plot structure provides alternative pathways through the representation of an overall plot related to a common story; the events, characters and settings of the story remain unchanged, but those narrated to the reader/viewer are interactively determined*
> *- a branching story structure involves interactive selection/determination of a representation of specific set of events, characters and settings constituting a story based upon a predefined set of potential events, characters and settings*
>
> (Lindley, 2005 [8])

In its most basic form these structures take the form of a branching tree. The story starts at a single point, the tree trunk, and specific choices encountered along the way will create at least two new versions of the story, thus creating new branches. By default, each branch has its own ending.

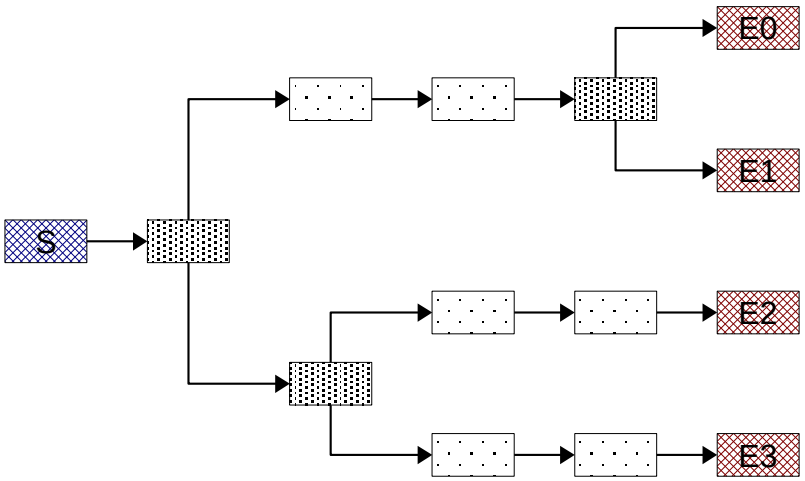


Figure 4: Simple branching tree

S : first scene/event (beginning)

E : last scene/event (ending)

: event

: event with choice

In this structure, there will be two choices to make. These choices will lead to the creation of four branches, and therefore to four different possible endings. They can either change the story, e.g. by changing the outcome of an event, or change the plot, e.g. by changing which character the player will follow.

However, adding more choices can make the number of branches grow very rapidly, a problem commonly referred to as *combinatorial explosion*. Each new branch requires new game assets, and inserts more opportunities for story inconsistencies. Therefore production costs, especially when compared to what amount of the narrative players will eventually experience in the game, get rapidly out of control and become unsustainable for the game developers.

To prevent this problem, designers introduce choke points in their structures in order to keep the number of branches under a manageable threshold. These are events shared by several branches, whose purpose it to regroup them into one, to reduce their number and facilitate production.

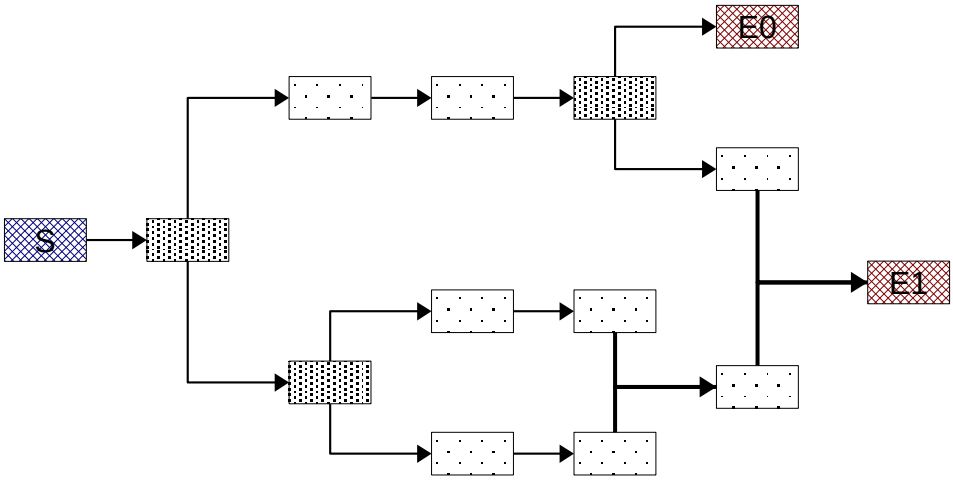


Figure 5: Tree with choke points

S : first scene/event (beginning)

E : last scene/event (ending)

: event

: event with choice

This structure is based on the one in Figure 4, However, here the number of endings has been reduced from four to two. The number of choices stays the same, which means reducing the amount of scenes to produce had a minimal impact on players' agency over the story.

A variation on branching trees that has some widespread use is what I choose to refer to as a hub structure. In it, branches represent series of events that are mostly independent from each other. They all start from the same level or event and, upon completion, bring back the player to that level or event. Players can choose to play through these branches in the order they want, and branches can be optional or mandatory.

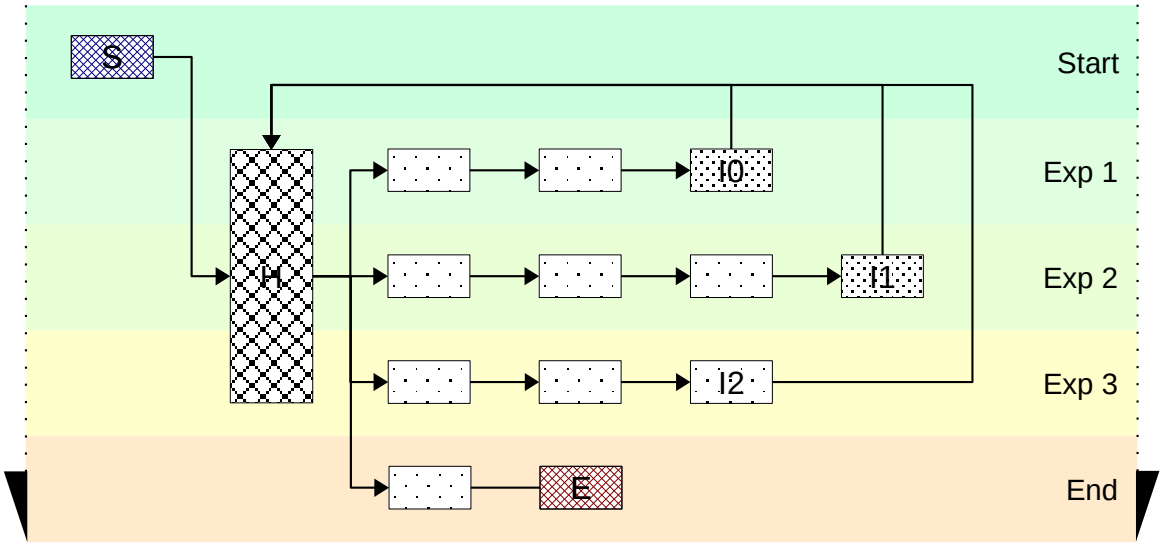


Figure 6: Hub structure

S : first scene/event (beginning)

E : last scene/event (ending)

: event

: required events to finish the game

: hub

Here, the plot starts with an event introducing the hub, after which 3 different branches can be played through. Two of them, I0 and I1, must be completed before being able to going through the final one, with the ending.

This kind of structure if most often used to simulate missions or expeditions launched from the same place, to show players the fate of different characters after one specific event (restarting at this event after finishing the story of each character), or to represents places that are all accessible from only one specific place (the hub). It can be part of another structure, for instance with the end of the hub structure leading to a “classic” branching tree structure afterwards. An example of it can be found in *Metal Gear Solid V* [9] : in this game the player has to manage a base, and their human resources, from which they can plan and launch missions in other parts of the game world, during which they can fill several objectives, some advancing the main story and some optional. It is also widely used in the roguelike genre.

All of these structure offer bases from which to build more complex structures, by combining them whether they’re from the “linear” or “non-linear” families. It is then up to the game developers to create structures to fit the stories they want to tell, according to their production capacity etc.

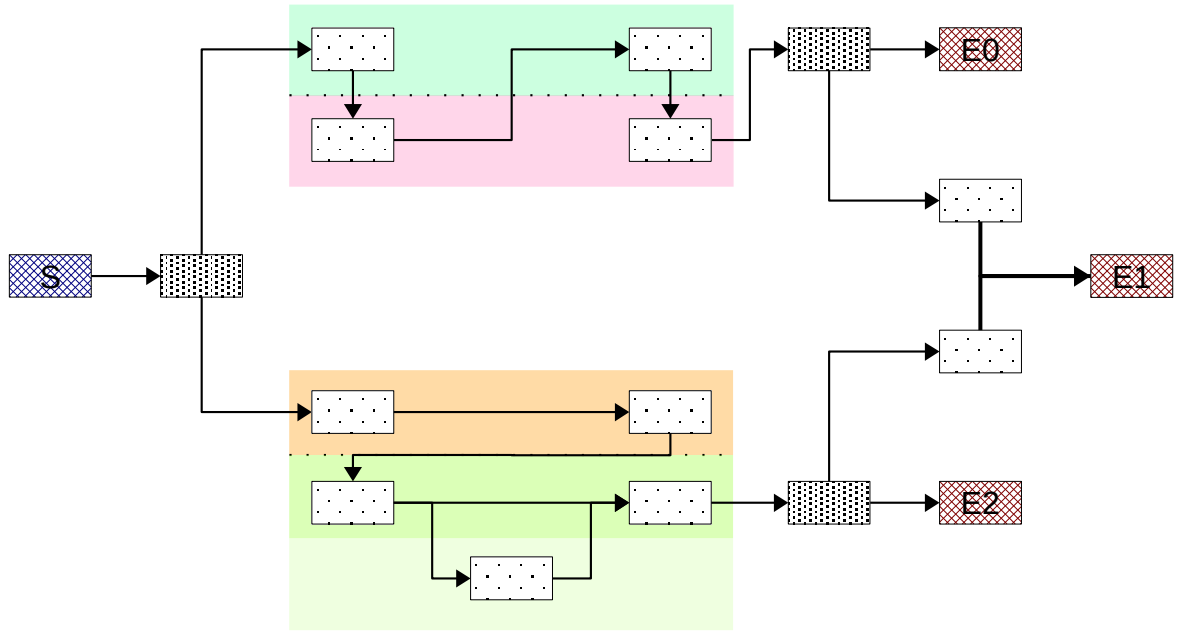


Figure 7: Combined structures

S : first scene/event (beginning)

E : last scene/event (ending)

: event

: event with choice

In this (relatively simple) example, the branches of a tree include other types of structures presented earlier. Both the top and bottom one have some form of parallel narration, and the bottom one even includes elements of elastic stories. This example represents a plot and story that go like this : the player is confronted with an important choice early on, which will determine how the core of the story continues, then they will follow the events lived by two characters in parallel, before being confronted with a second choice to determine how the relationship between these two characters ends, either as enemies (E0 and E2), or as friends (E1).

The underlining narrative structures of games can be hidden or not. When openly shown, they can be used as a narrative device to confront the player with their playing style and previous choices, their position toward the events unfolding or toward the characters attitudes. Openly admitted choices are the main selling-point of many role-playing games, and it is reflected in how they're presented to the player. In the *Mass Effect* series [10], the choices are presented next to each other, and some choices are colored blue or red, to indicate whether they are considered "morally good" or "morally bad" in a straightforwardly way (see *Appendix 1*). It can go even further by completely showing the structure to players, as in *Detroit: Become Human* [11], in which each chapter ends with a flowchart showing all the branches the player went through and all the branches they didn't use, and might have gotten had they chose other actions (see *Appendix 2*). Alternatively, some games hide their structures and resolve narrative choices with actions the player made elsewhere or with other, seemingly unrelated choices. For example in a game like *Fallout: New Vegas* [12], some quest choices are dependent on the level of certain stats your character has and his or her relationship with the factions of the game world, which means that according to the actions the player made before, some choices will auto-resolve or simply be unavailable.

Ultimately, these structures are here to give the player freedom and giving them agency upon the game world, letting them choose the narrative knowingly or not. The final game experience is linear however, *« the player (consciously or unconsciously) creates one series of events from several possible alternatives, thus actualizing one of several latent stories »* (Backe, 2012) [13]. Non-linearity is there to validate the choices, because without it they wouldn't have any meaning and the player wouldn't have any real agency on the game world. The only way for a player to actually experience non-linearity would be to play through a game multiple times, playing differently and making different choices on purpose.

Open worlds

Among the games using non-linear narrative structures, open worlds stand out as a singular case. Currently a staple of the blockbuster video game industry, open worlds include all forms of narrative structure, but set them in the context of an open space players can wander in and explore freely. Their organisation ranges from simply being a bigger space to house a string-of-pearls model to scattering out narrative elements all over the game world without necessarily linking them with a bigger narrative. In the latter case, the player comes across them and learns about the history of the game universe only by moving their avatar in it, in an approach which, as Henry Jenkins puts it, examines *games less as stories than as spaces ripe with narrative possibility* [14].

Independently from the structures used, open worlds play with the idea of complete freedom of movement. As they allow players to wander a big space freely, without interruptions, and come across narrative elements, willfully or by chance, open worlds are

theoretically the less linear experiences available currently in video games. In practice though, the narrative usually centers around one main structure, presenting the story of the biggest events and main character, along with several secondary structures with various purposes. This is because in most cases the game developers want to tell one specific story while also using the possibilities of the open world to tell optional secondary stories, or fill the game space with activities so it doesn't feel empty. Separating and hierarchising structures allows game developers to add causal dependencies between events to ensure the chronological coherence of the plot, and avoid side-effects of the liberty of movement given to the player. Elements of game design and gameplay systems are used to guarantee some degree of linearity in the experience of the narrative, such as markers showing where to go next to continue the main storyline, or difficulty scales that make some areas of the game world harder to explore than others (e.g. by putting harder enemies in it), forcing players to explore the easier zones first, in an order decided beforehand and corresponding to the plot order.

On the other hand, other elements of game design and gameplay systems in open world games exist solely to force the player to follow the narrative non-linearly, by making it impossible or extremely difficult. Examples include making the main storyline quests' difficulty non-linear, to force players to complete secondary quests in-between main quests in order to make their avatar stronger, or asking the player to collect resources in other activities to be able to progress in the completion of an objective on the current plot branch. Another frequent technique is dividing the storyline into parallel narrative trees, representing different parts of the story happening "at the same time". It appears for instance in *Fallout: New Vegas* [12] in which different factions each have their own plot arc, the player being able to advance in them independently, only one of those arc needing to be completed to finish the game. These narrative trees can be completely disconnected from each other.

A common narrative device in role-playing games is to reveal the end goal of the story directly at the beginning, to start an adventure following the hero's journey template. All the content of the game is then seen as challenges or obstacles to overcome in order to finally be able to achieve this end goal. Some designers have recently used this template in combination with the open world structure to get rid as much as possible of intermediate parts in the story, and make the world itself the challenge. A famous example would be *The Legend of Zelda: Breath of the Wild* [15], in which the player is given their final objective very early after the game starts, and no game system prevents them to go and accomplish their goal directly. In this game, the open space around the only area the player has to complete is entirely optional, event though it constitutes the core of the game experience, the body of the main character's journey. This approach is trying to make the most use of what constitutes an open world, i.e. the game space, and use it directly as a narrative device driving the story.

Open world games are an attempt at maximum narrative liberty for the player that uses one of the specific features of video games – being able to move around in a virtual space – to create a stronger degree of agency. They therefore occupy a specific place in game development, and make use of "tricks" to make the use of already existing narrative structures possible in order to keep the production of such games feasible. However, their very nature means they tend to push the boundaries of narrative structures, as these structure have to be non-linear enough to use the game space, and open new ways to tell stories via game systems.

Narration in video games with open, non-linear stories is a construction between the author(s), who create the structure and events so the player is able to experience it, and the player(s), who by going through the game is actually creating the pace, timing and order of the events, thus creating the final plot or story. Unlike in some other media, where the readers/spectators/users encounter finite experiences (i.e. experiences where the full sequence of events is set before the beginning of the experience), the work of video games authors ends before the narration is finished. The final narration is made by the player, as they assemble the elements of gameplay and narrative to form their own specific story. Video game design and production tends, in some cases, toward including the player as an agent of the narrative from the start, by creating systems capable of reacting to players actions or to allow them to make choices non-explicitly, by playing the game.

Narrative gameplay systems

Authorship in video games with non-linear narratives has been moved away from just creating, explaining and showing a sequence of events directly. It has to take into account the specific features of the medium, such as how a story's pace is modified by the gameplay activities the players have to go through. It means the narration has to take into account and include the act of playing itself, using the unique features of the medium to tell a story, which leads to bridging game systems and narrative structures. It ranges from creating several sequences of gameplay, each showing a different part of the plot, to creating systems able to track a players' progression in the game and to adapt the plot or story to their progression. Narrative systems became so important that it lead to the emergence of a new category of game developers in the industry : narrative designers, whose role is not only to create narrative content, as writers do, but also, and in some cases exclusively, to design *ways* to experience this content. These often comes down to finding how to tell stories with – or through – systems.

Proactive systems

The most obvious kinds of systems are the explicit ones, those that often take the center stage in a game's marketing and around which the games are built. Marketing material often refers to these systems as *reactive* [16] [17], and explain the player may play however they like and the game environment will react to their actions. In truth these systems are more akin to something that should be defined as *proactive*. They are composed of multiple subsystems, some of them interacting with specific player actions and some interacting with other subsystems. The extent of interactions the system can interpret is limited, and therefore can be considered planned in advance. Thus, players actions are anticipated, not reacted to. The apparent reactive nature of the game system is due to the number of interactions anticipated, and to the interaction between subsystems, which creates emergent paths and ways to play. The designers anticipated how each activity and system can interact with the player and other systems, but not necessarily how they could "chain-react" with each other.

The sandbox model, in which a finite game space is filled with activities, AI life and objects to interact with, is a great – and explicit – example of this kind of system. Games such as *Minecraft* [18] or *Rust* [19] feature open spaces, systems such as crafting to create objects, resource gathering and building, and make all these available without giving players clear goals. They are given things to do, but no explicit motivation. The narrative is chosen by the player when they decide what to build, where to gather resources, which part of the world to explore, etc. The narration is created in real time by the player when they use the systems in-game, there is no need for a pre-written story. This is often exacerbated by the online nature of some of these games, which means players can also create their narrative by interacting with each other.

Immersive sims are a specific genre of games with ties to sandboxes. They also feature high player agency and the simulation of a believable space, but are much more story-driven. In it, the players are told what to do but not how to do it, and will have to pick their own gameplay styles and routes toward the given objective [20]. In *Dishonored* [21], one of the most famous series of game in the new (around 2012) wave of immersive sims, the player is given detailed goals - "Kill person X or Y", "Retrieve object Z" -, some equipment,

a basic set of actions (look, move, attack, shoot, jump), and is then left to their own device in a delimited area filled with different types of NPCs and short quests. It is then up to the person playing to figure out and set their own intermediate goal, like finding out how to get access to the assassination's target apartment, set up a deadly trap, distract guards and so on, to help them achieve the end goal. Along the way, several hidden end goals will be unlocked and may change the course of action they initially chose. While the idea is to offer players a play area as open as possible, the goals can only be achieved in a finite number of ways. The more a path involves narration and specific actions, the more limited it is. The system is not reacting and creating a new route to the objective based on players actions, it is making available an already existing path and triggering events when players do the necessary sequence of actions or the right choices.

In these examples, the narrative comes from the system, with which the player will determine the order of events and which events occur or not. It is not up to the game itself (to various degrees) to determine what *will* happen, but what *could* happen.

Reactive systems

On the other hand, there exist game systems that take an opposite approach. Less visible, because of their hidden nature, their goal is to create, or at least change, the narrative during play. These systems have the ability to create or modify other game systems; they follow sets of instructions but track players' input to know what to do. As such, they can be called *reactive*, contrasting with the systems presented earlier. Such systems have been implemented and theorized very early in the development of game design, as early as the 1980's [22]. One of the first examples of an adaptive system is what's refereed to as *dynamic game difficulty balancing*, systems aimed at making sure a game is never too easy or too hard by modifying enemies' strength, number, etc. Other systems followed, building upon it, around the same idea of managing the pace and difficulty of a game over time, in order to create an "optimal" game experience for the player. With time and newer technologies, game spaces increased in size and density, and new game systems were designed to answer the problem of populating these spaces, to save production time and/or to create a better feeling of immersion while going through the game world.

The idea of systems emulating the role of a tabletop role-playing game *game master* has been present in the video games industry for a long time, because video game RPGs originate from these tabletop games. The role of a game master is to manage the universe and quests made available to players, react to their actions and orient them toward the narrative they created. Such systems have been implemented in various ways, but a relatively recent and famous one is *Left 4 Dead*'s [23] AI Director. The game is a 4-player cooperation game where a team of players has to work together to go through a level, surviving waves of zombies along the way. The AI Director works by computing the stress level of individual players, their progress in the level and their field of view to adapt the number and strength of the enemies encountered to create a desired pace, with unpredictable moments of heavy intensity following longer moments of calm [24]. It also reacts to the characters' positions relative to each other and their equipment, e.g. to take advantage of a weak player's isolation to put them in danger, forcing the rest of the team to come to their rescue. It benefits from the limitations of the game (linearity of the levels, little narrative content) and the strengths of its production (years of game AI research to rely on). This system has several advantages: it guarantees an experience corresponding to the intentions of the designers whatever the player team's composition and it creates

emergent narrative through gameplay mechanics, efficiently instilling specific emotions in players without relying on external assets like text or voice acting. It has, however, fairly high production requirements - "*they were built on years of Counter-Strike bot navigation work, hundreds of motion captured animations from professional stunt people, and years of additional work"* [25] - even though it is still cheaper than solutions bigger studios might have chosen - "*If we were a larger studio, perhaps another solution would be to build many maps, or many, many variants of populations within a map."* [25].

Games that give player large open spaces to experience, especially open world role-playing games, suffer from an inherent problem: emptiness. Between the places hand-filled with content by level designers lie vast spaces the player has to go through. If these spaces are empty and don't lead to interactions often enough, it can hinder the game experience. This is one of the motivations behind secondary content, such as quests not linked to the main story, mini games, craft systems, etc. These elements of gameplay are not rich in narrative content, if they have any. Systems have been developed to address this problem and create new narrative in gameplay areas otherwise deprived of it. A prime example of such systems is the Nemesis System developed for the *Middle Earth* [26] series. It was created to add narration in encounters with "trash mobs" (enemies found in the open space, not linked with any mission or quest), and at the same time to link these to the main gameplay content. In the games, if the player avatar is defeated by a mob it makes this normally standard, unnamed enemy a Nemesis of the player, giving it a name, an improved graphic design and making it harder to beat afterward. Over time the player can come across and fight this enemy multiple times, and even eventually recruit them as a companion. It creates a stronger sense of belonging to the game world, as any basic enemy feels like a "real" entity able to influence and be influenced by the player. It is however limited by the fact it relies on pre-written lines for dubbing and sometimes on specific versions of 3D models for the Nemeses, but its strength is that any basic enemy can become a Nemesis, as they are generated randomly by combining systemic character traits (like sets of attacks, vocabulary, where they can be found), weapons and 3D models [27].

Contrary to the proactive ones, these systems act to create narrative and make it emerge from gameplay. The exact sequence of what could happen is not managed, to instill a sense of discovery in the player and hide the systems as much as possible during their playthrough, but what will happen - like where will the player encounter a horde of zombies in *Left 4 Dead* - is decided in real time depending on what players do.

Narration in role-playing games, or at least in games with direct control of an avatar, is build for/around the player, who is a subject of the narrative : they choose their own actions, unlike a passive hero waiting for their fate. Many other genre outside of the ones including role-playing elements and direct control of an avatar, such as simulations, management and strategy games, include narrative systems or gameplay systems that create narratives. They are not part of the scope of this thesis, and have been left out on purpose..

The integration of narrative and gameplay

Although sometimes studied separately, narration and gameplay are not opposed. While the industry chose for the most part to split the production of games between these two categories, mostly by creating independent teams or positions (like game designer and narrative designer), and the dichotomy was kept alive by the “ludology vs. narratology” debate, these are not completely separate concepts. Having always fed each other during production and worked together to provide the final game experience int the subset of games of interest to this thesis, the problems found between those concepts often emerge from problems with their integration.

Room for improvement

The evolution – and improvement – of narrative structures and systems leads to the improvement of video games stories, plots and narration. But they also put more pressure on game studios, as they introduce more requirements, more details and layers of complexity that take time to develop, and make the management of a production more laborious. As much as they solve problems, they may also bring their own lot of problems with them. These systems are all about improving creativity while balancing the players satisfaction and the limits of the production, ideally to avoid having negative repercussions on the working conditions of the studios employees and economical health.

During most role-playing games production, the creation of the global narrative outline precedes the game design. Moreover, the plot and story tend to not change much, if at all, during the game’s development, because the nature of this genre (putting the player in the role of a character in a story) means they’re created from the start as a way to tell a story, or at least to recreate the experience of being a hero in a story. The systems developed later are then either not directly related to the narrative, or are created to serve it. In organisational terms, it translates as narrative and gameplay components of a game being often produced in parallel. With different teams working on each of these parts, it comes down to the management of the respective teams to ensure they communicate well. It can lead to issues with both how each team understands the others’ work, and miscommunication introducing delays in production as people scramble to solve the emerging problems (bugs, inconsistencies, etc.).

Different studios have different solutions, none being perfect, to solve these complications. While bigger ones will try to have an efficient production team dedicated to managing communication and schedules between the other departments and a QA team working to find bugs as early as possible, smaller studios may choose to alternate between designing narrative and gameplay features, such as starting by putting in place some narrative, then adapting the mechanics to it, going back to work on the narrative and changing it to fit their progress on gameplay, and so on. A common result of bad coordination arises when the game starts being playtested : if feedbacks suggest a narrative arc or a gameplay feature critical to the narrative is not interesting enough to players, it will hardly be corrected at such a late stage of development, and inconsistencies will be hard to correct. In the most extreme case, these will lead to some content being simply deleted from the game, as it might easier and/or cheaper to do so.

From the player side, games can sometimes feel imbalanced, as if gameplay was subordinated to narrative design or the other way around. It can sometimes be intended as designers wanted to focus, were more interested in, or were better at creating one of the aspects.

Narrative systems are difficult to hide, and their presence can be made too obvious. For instance, non-playable characters can be blocked in a state where they're waiting for player input to continue their routine or story briefs can talk about "an urgent task to accomplish" but have the player free to take their time and do whatever they wish to do before actually accomplishing this task. Systems can also be detrimental to narrative freedom, as in the example of *Left 4 Dead*'s AI Director, which exists and works very well, but to the expense of spatial freedom, the levels having been made smaller to allow the system to be efficient. Another common problem are the false causal relations, with a common example : a player goes into an empty cave, then is later given a quest to slay a dragon, which happens to be in the same cave. Why was the dragon absent before ? The game system waited to spawn the dragon after the quest had been started, which leads to an illogical situation.

Narrative structures suffer from the same problem. Players may feel that secondary missions have been added as filling in the environment, which makes them appear unnecessary, without consequences or impact. Inconsistencies left out in the final product will get the player out of their immersion, reminding them they're plating a game, not "living" an adventure.

These problems lare of course not common to every video game, but they tend to represent standard occurences for role-playing games with a strong story component. They lead to a feeling of ludonarrative dissonance when playing the game, a feeling that what is told by the game doesn't correspond to what players actually do in it. These conflicts in the integration of narrative and gameplay in quests may be solved (at least partially) by looking at them a different way, and adapting their structures and production.

A quest template : finite-states machines

In the Arthurian legend, the quest for the Holy Grail is a tale of heroes going on a journey to achieve great feats. The quest itself is the journey, the act of seeking to achieve a goal. This concept has been reused in tabletop role-playing games to give an objective to players, and to make the adventures they go on more interesting. In modern computer role-playing games, it can be simply defined as *an activity in which players must overcome challenges in order to reach a goal* [28]. They are an important concept in video game design, widespread, well-known and well-understood, which makes them a strong basis on which to build a game structure. Their versatility and their isolated nature are very useful. Quests act as *a conceptual bridge that can help to join together many pairs that are often considered separately* [28] – in our case gameplay and narrative – in a single, encapsulated unit.

Template

Traditionally quests are a series of event, part of a tree or not. These events happen one after the other, and await player actions to progress. Let's start with an example of a quest : in this example, after some events in the narrative led both to its impoverishment

and to the raise of taxes, a peasant revolt is ongoing in region A of the game world. Upon arrival in this region, the player character meets the Leader of the revolt, who asks them to help by finding the Lord of the region and killing him, so the revolt can succeed. If the player does that, the Leader will reward them. This quest can be represented this way :

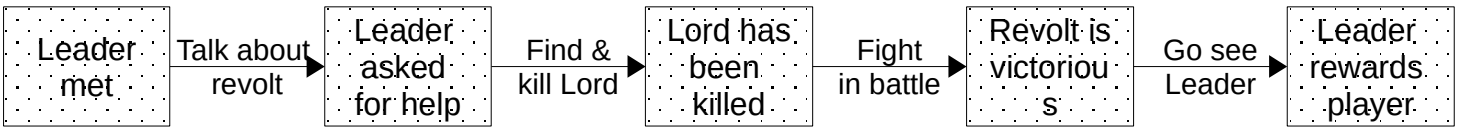


Figure 8: Basic quest example

This representation shows that this quest can be considered, in fact, as a Finite-State Machine (FSM). Indeed, the quest can only be in one state at any given time : if the state is “Lord has been killed”, it means the player found and killed him after the Leader asked for help, but the battle hasn’t occurred yet.

This kind of construction for a quest poses a problem : what if the player does nothing ? In this case, the state of the quest (and therefore, of the game world) doesn’t change, this part of the plot is stuck, seemingly frozen in time, until the player does something. Causal relations can be a problem too, and may add many opportunities for incoherent narration : what if the player kills the Lord before having met the Leader ? Or if they kill the Leader ? Solutions like failing the quest automatically or blocking the player from doing certain actions (e.g. by spawning the Lord in the castle only after having met the Leader) are unsatisfying narratively and limit players’ freedom.

I’ll present here one of many possible solutions to some of these problems. The base of this template is to consider quests not as a single Finite-State Machine, but as a collection of several FSMs managed by the quest’s program, that uses and updates state machines and variables found in other components of the game (non-playable characters, areas, etc.).

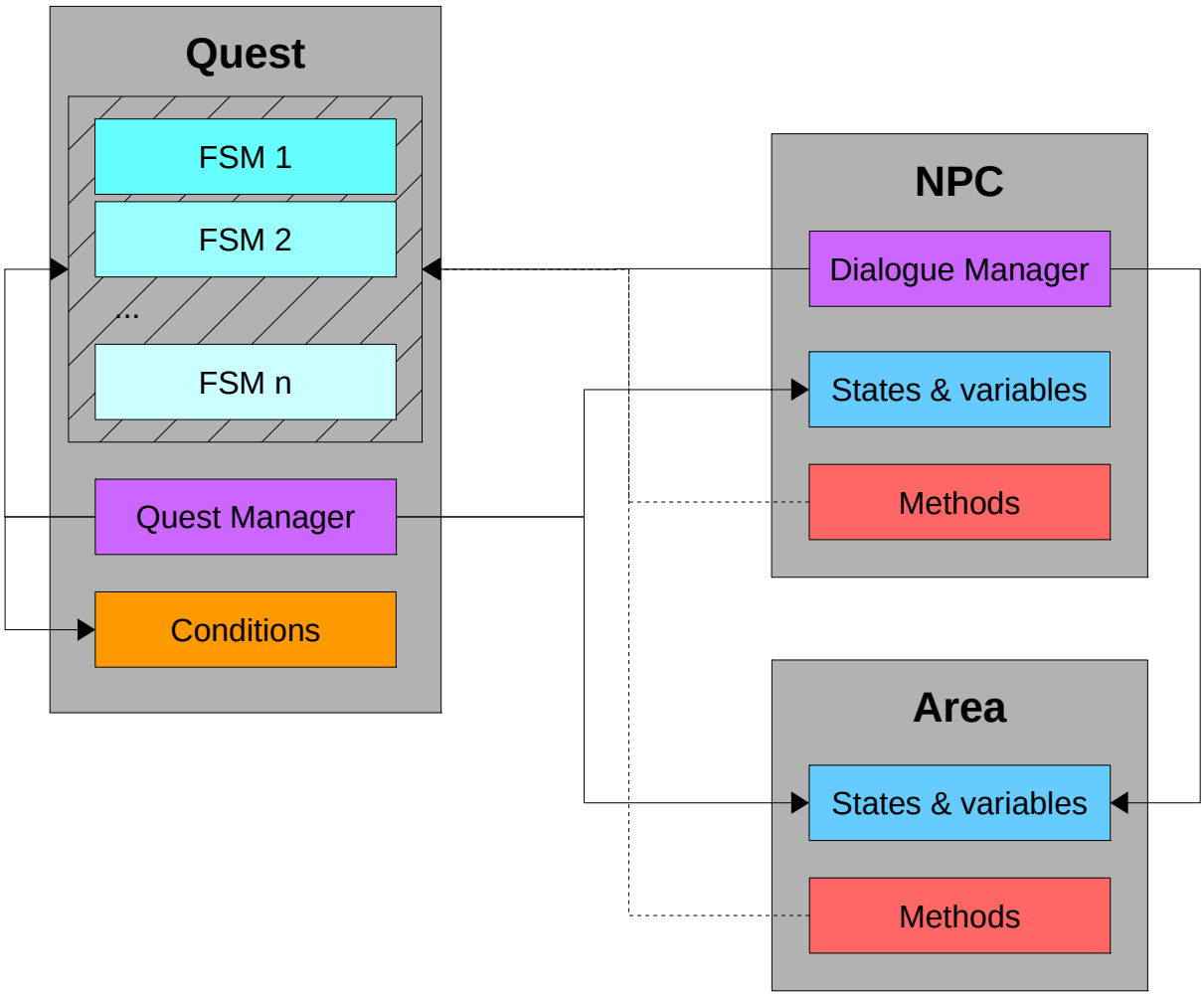


Figure 9: Quests and relations to other components

The figure represents one quest in relation with one NPC and one area. In reality, quests can be linked to other components as well, such as objects, skill trees, etc.

The **quest manager** centralises all the logic (tests, updates) of the quest. It is responsible for starting the quest, which it does by comparing the **conditions** and the states and variables of other game components. It is also responsible for updating the quest's **finite-state machines**, which, combined together, represent all the possible states of the quest progression. It uses information from other game components' **states and variables** to do so.

NPCs' **dialogue managers** contain all the lines of dialogue of the NPC, and all the conditons and tests needed to know when and how to display these lines of dialogue in the dialogue system. These tests check against states and variables from the other components. Under some circumstances, the dialogue manager may also update states in the quest's FSMs.

To create a quest, a game developer has to decompose what they want to happen in it, and what they want the player to do, in several finite-state machines. They then put down the conditions necessary for the quest to be able to begin, and create the logic that will update and advance through the states, i.e. the quest manager. Finally, they can add the

lines of dialogues, and their conditions, to the NPCs' (those used in the quest) dialogue managers.

Example

Let's illustrate the template by remaking the precedent quest example with it. We will start by the decomposition of the quest in several FSMs. The main event of the quest is the peasant revolt; we want to decompose it for dramatic purpose in several steps : the beginning, with peasants being angry and refusing to pay taxes, the peak of the revolt, when they fight the Lord's men and chase tax collectors, and a final battle at the Lord's castle. After this battle, the revolt will either succed or fail, depending on the states of the quest (and therefore, depending on players actions).

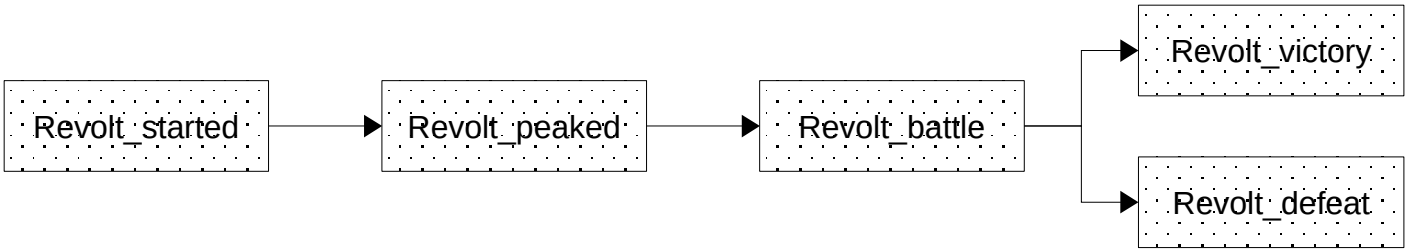


Figure 10: Revolt FSM

The Leader may have asked the help of the player, after which the player may have found the location of the Lord and then told the Leader about this location. Later, if the Lord is dead the Leader will be happy, but if the revolt has succeeded without the help of the player, as was agreed, he will be resentful of them.

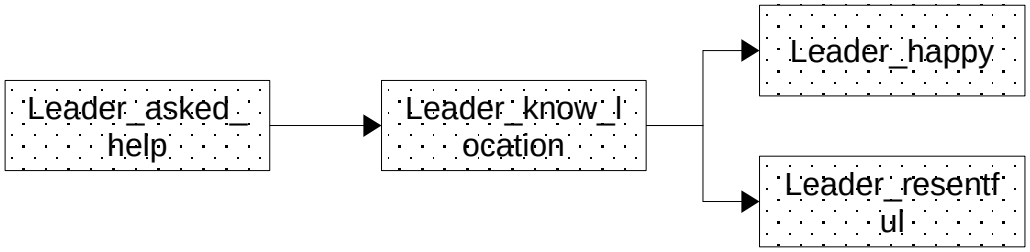


Figure 11: Leader FSM

Finally, the Lord can be either hidden (no specific state, as it is the default), located, or killed.

Figure 12: Lord FSM

With these three finite-state machines, all the wanted states of the quest are covered. It is important to note here that the states in these machines imply the states preceding them: if the manager moves the state to *Revolt_battle*, it implies that *Revolt_peaked* and *Revolt_started* have been active before. This is important for the coherence of the tests in the quest and dialogue managers. Next the conditions of the quest have to be set. The

narrative setting needs region A to be poor, taxes to be high, and of course the Lord and Leader present in the quest must be alive at the start. The conditions are then :

- *Region_wealth* (variable of the region component) must be equal to poor;
- *Region_taxes* (variable of the region component) must be equal to high
- *Lord_alive* (a variable of the Lord NPC) must be true
- *Leader_alive* (a variable of the Leader NPC) must be true

We also want the quest to evolve in real time, so it becomes a dynamic event which doesn't wait for the player to act. This is why we create a variable *quest_timer* to track the number of days since the beginning of the quest. When all the condtions are met, the quest manager starts the quest timer, and moves the state of the Revolt FSM to *Revolt_started*.

The quest manager hosts all the logic of the quest. It will advance through the steps of the revolt depending on how much time has passed since the beginning of the quest and, when reaching the final day, will determine the outcome of the revolt (and therefore, of the quest) depending on what happened since the beginning.

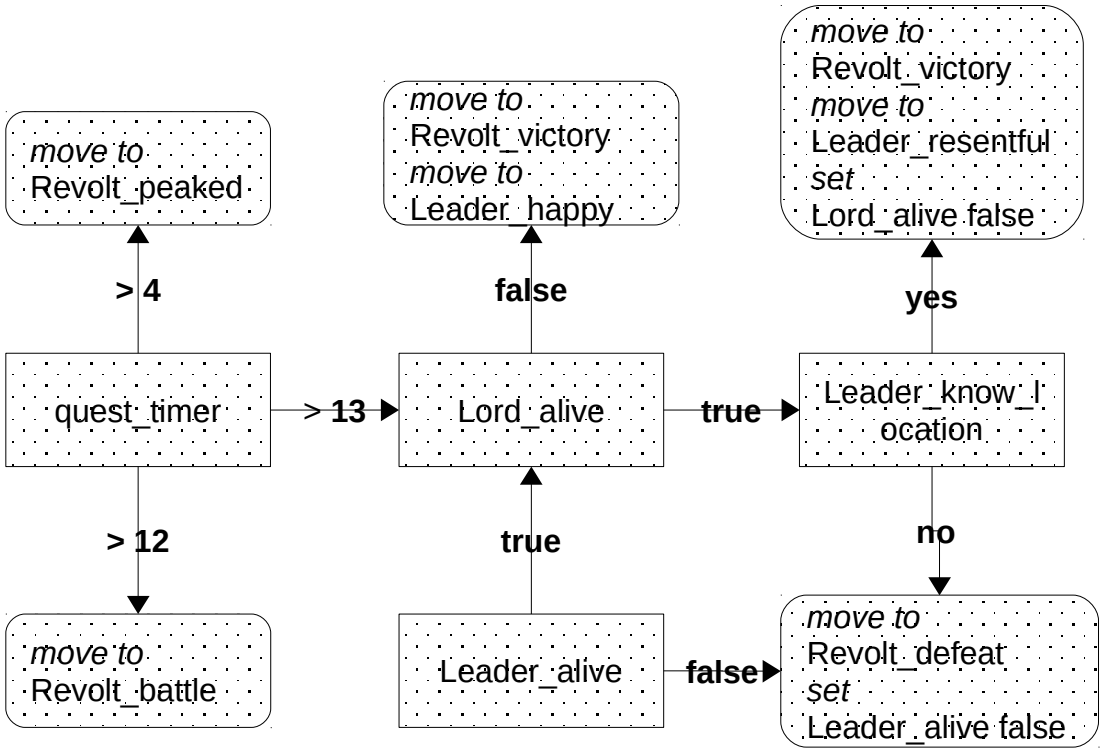


Figure 13: Quest manager logic for the progression of the quest

In this example, the revolt's stage moves forward after 4, 12 and 13 days. At the end (after the battle), its outcome depends on whether the Lord and Leader are still alive or not, and if the Leader knows the whereabouts of the Lord (if the player came back to tell him). In case of success, the Leader will reward the player differently, depending on whether the Lord was killed by the player or by the peasants themselves.

The quest manager will also initiate changes to game components used in the quest. For instance, when moving the state to *Revolt_peaked*, it will also change the state of the region, which will enact changes in it : add more bandits in it, activate random skirmishes between peasants and soldiers, make a village burn, etc. The quest timer is incremented

every game day by the quest manager itself, and other states are set by external components, like *Leader_know_location* being set by the dialogue manager of the Leader.

With this system, the player has more freedom. Many game scenarii are available to them, including the same one as in the example where the quest is made up of only one finite-state machine, and we can imagine some of those :

- *the player never goes in region A* : in this case, the revolt will auto-resolve, and result in a defeat. The Leader NPC will be unavailable, and people in the region will be able to talk about the revolt later on;
- *player kills the leader in another quest, during the revolt* : here, the quest will end with the defeat of the revolt;
- *player has killed the leader long before the beginning of the quest* : in this case, the quest will never even start.

Now let's see a slightly more complicated scenario : the player kills the Lord then meets the Leader after the revolt has ended. In this scenario the revolt will auto-resolve to a victory, and the Leader will be happy. The end states will be like so :

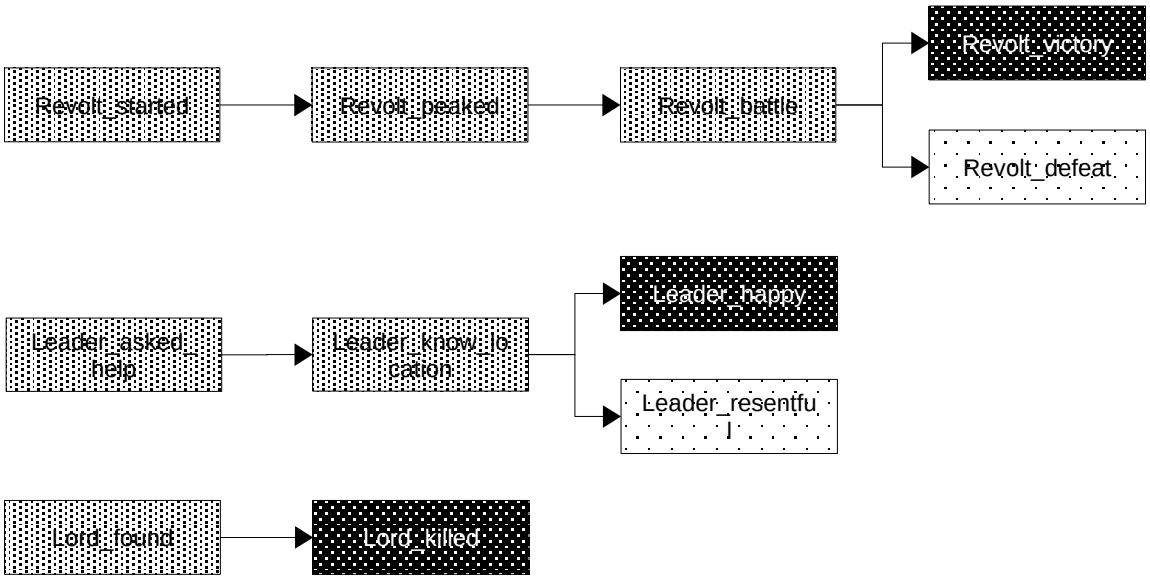


Figure 14: States at the end of the scenario

This makes it possible to identify a problem easily : if the Leader is happy and rewards the player, it means the state *Leader_asked_help* as been activated before, which is actually not the case here, since the Leader hasn't even been met. This can be solved easily, by adding some dialogue to the Leader's dialogue manager, in order to manage this specific case, and modifying the logic quest after the test on *Lord_alive* :

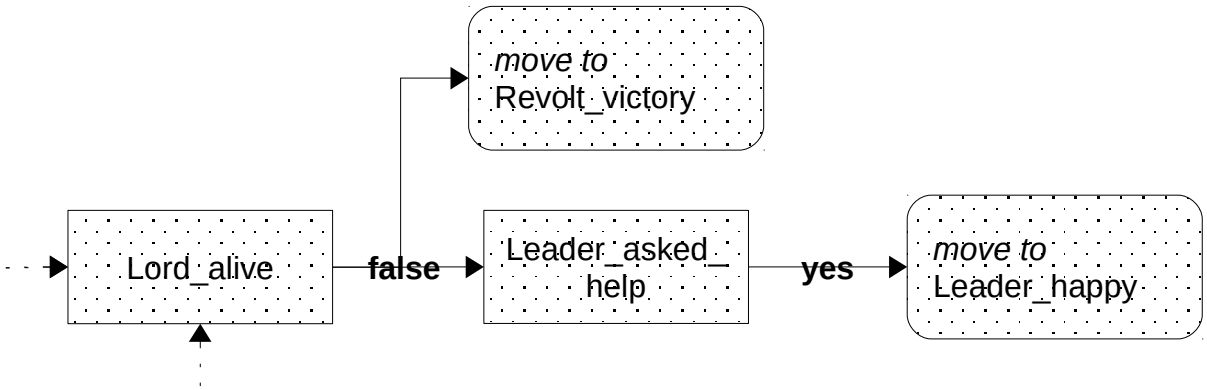


Figure 15: Corrected branch of the quest manager

Benefits and downsides

With a template like this one, editing quests separately is easier. They don't necessarily have to rely on each other, or a bigger narrative structure for that matter. It makes it easier to edit, manage and modify quests without breaking the entire structure, while still making it possible to write dialogue (for quests or for atmosphere) relying on events from quests already completed. The data contained in finite-state machines is therefore highly reusable. The template's basis makes it easy to add default states and default lines of dialogues which prevent players from getting stuck. Other advantages include how it can be easily represented using diagrams and drawings, helping with communication among the production teams, and the ability of quests to be broken down into modules (game components, FSMs, etc.) which can be reused or maintained separately.

In theory, if the quests' FSMs and managers are carefully made the template is really efficient and is able to catch and manage any unforeseen combination of states, which means the consistency of the story will be really strong. But in reality, and especially within usual production constraints, I suspect this template would result in more bugs because of the sheer amount of combinations possible, which could grow exponentially in a bigger game. However, since most of those bugs would be the result of unknown or unplanned combination of states, fixes could be made quickly and easily by narrative designers (or anyone responsible for the writing of the quests) : a few lines of dialogues for the relevant NPCs, with tests corresponding to the problematic combination, and some changes to the logic of the quest manager (as seen in the example) would be enough in most cases.

One limitation is the amount of redundancy in the dialogues. With many possible combinations of states, there will be lines of dialogue close in meaning but with slightly different content. This can be cheap to produce however, if it only amounts to text that is not voice-acted. This is why this template appears to be more suited to text-based games with a significant amount of dialogues. The template is also used optimally with part of the developing team specifically working on tools to create software for the visualisation of the quests and the associated game components.

Examples exist in the literature that bear similarities with the template presented in this thesis, such as object oriented story construction [29], which relocates all the narrative content and logic and attaches it to the game objects themselves, or the systems used successfully by Inkle Studios in their games [30], The latter ones have been used in

several critically acclaimed video games, with people at Inkle acknowledging the role of the template they used to drive the narrative in these successes. In all cases, these concepts have been developed to try to solve similar problems and improve production. I’m therefore confident in the fact that such templates are and will be useful, and that their development will be beneficial overall to narrative-focused games.

Conclusion

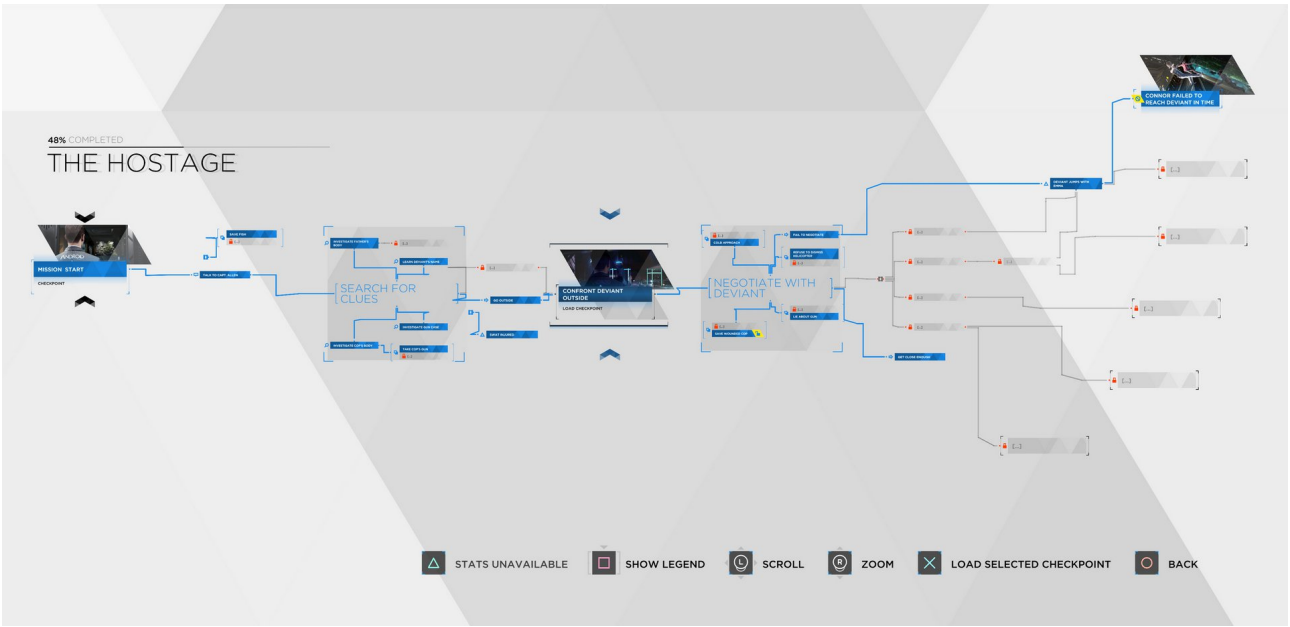
Narration in video games has the particularity of being the feat of both the game developers and the players. The interactive nature of video games is what draws many people to play them, and to satisfy this games have to include specific features like narrative structures and systems to offer players agency over the story. These exist in various shapes or forms, depending on each game creator's goal. Some problems have been identified in role-playing games using quests as gameplay and narrative units. I proposed a template for quests, based on my experiences as both a player and a game developer, to try to answer some of these problems.

This template will require more work, and most importantly will need to be implemented and tested in an actual video game prototype, to test the process of creating and integrating quests and to ensure it is meeting its goals. Later on, I would like to expand it by using Smith's *Quest Pattern Library* [25] to create new dependencies between game components and automatically track quests' types to adjust the player's interactions according to it.

In the more distant future I think this improved system would also be a good candidate to be combined with quest generators, like the ones developed by Smith et al. [32], or Kybartas and Verbrugge [33], in order to try to create an engine for the procedural generation of quests.

Appendix

Appendix 1: Mass Effect 3 dialogue system



Appendix 2: Detroit: Become Human end chapter flowchart

Bibliography / Ludography

[1] G. Genette, *Figures III (Discours du récit: Essai de méthode)*. Éditions du Seuil, 1972.

[2] Naughty Dog, *Uncharted (series)*. Sony Interactive Entertainment, 2007.

[3] D. Cage, “Postmortem: Indigo Prophecy,” *Gamasutra*, 20-Jun-2006. [Online]. Available: https://www.gamasutra.com/view/feature/131140/postmortem_indigo_prophecy.php. [Accessed: 11-Jan-2019].

[4] Quantic Dream, *Heavy Rain*. Sony Computer Entertainment, 2010.

[5] Telltale Games, *The Walking Dead (series)*. Telltale Games, 2007.

[6] Quantic Dream, *Fahrenheit*. Atari, Inc., 2005.

[7] R. Činčera, *Kinoautomat: Člověk a jeho dům*. 1967.

[8] C. A. Lindley, “Story and Narrative Structures in Computer Games,” p. 27, 2005.

[9] Kojima Productions, *Metal Gear Solid V : The Phantom Pain*. Konami, 2015.

[10] Bioware, *Mass Effect (series)*. Microsoft Game Studios, Electronic Arts, 2007.

[11] Quantic Dream, *Detroit: Become Human*. Sony Interactive Entertainment, 2018.

[12] Obsidian Entertainment, *Fallout: New Vegas*. Bethesda Softworks, 2010.

[13] H.-J. Backe, “Narrative rules? Story logic and the structures of games,” *Literary and Linguistic Computing*, vol. 27, no. 3, pp. 243–260, Sep. 2012.

[14] H. Jenkins, “Game Design as Narrative Architecture,” in *First Person: New Media as Story, Performance, and Game*, Noah Wardrip-Fruin and Pat Harrigan., Cambridge: The MIT Press, 2004, pp. 118–130.

[15] Nintendo EPD, *The Legend of Zelda: Breath of the Wild*. Nintendo, 2017.

[16] S. Petite, “BioShock Creator Says His Next Game Will Have an Ambitious, Reactive Narrative,” *Digital Trends*, 30-Mar-2017. [Online]. Available: https://www.digitaltrends.com/gaming/ken-levine-next-game-infuences/. [Accessed: 12-Jan-2019].

[17] N. Grayson, “Wasteland 2’s Delay: All About Making Choice Matter,” *Rock, Paper, Shotgun*, 30-Jul-2013. [Online]. Available: https://www.rockpapershotgun.com/2013/07/30/wasteland-2s-delay-all-about-making-choice-matter/. [Accessed: 12-Jan-2019].

[18] Mojang, *Minecraft*. Mojang, 2011.

[19] Facepunch Studios, *Rust*. Facepunch Studios, 2018.

[20] M. Brown, *The Comeback of the Immersive Sim*. 2016.

[21] Arkane Studios, *Dishonored*. ZeniMax Media, 2012.

[22] C. Crawford, “Design Techniques and Ideals for Computer Games,” *Byte Magazine*, vol. 7, no. 12, Dec-1982.

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

[24] M. Booth, “The AI Systems of Left 4 Dead,” presented at the Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford University, 2009.

[25] M. Booth, “Mike Booth, the Architect of Left 4 Dead’s AI Director, Explains Why It’s So Bloody Good,” *Kotaku Australia*, 21-Nov-2018. [Online]. Available: https://www.kotaku.com.au/2018/11/mike-booth-the-architect-of-left-4-deads-ai-director-explains-why-its-so-bloody-good/. [Accessed: 11-Jan-2019].

[26] Monolith Productions, *Middle-earth: Shadow of Mordor*. Warner Bros. Interactive Entertainment, 2014.

[27] C. Hoge, “Helping Players Hate (or Love) Their Nemesis,” presented at the GDC, 2018.

[28] J. Howard, *Quests: design, theory, and history in games and narratives*. Wellesley, Mass: A.K. Peters, 2008.

[29] M. Eladhari, “Object Oriented Story Construction in Story Driven Computer Games,” p. 101, 2002.

[30] J. Ingold, “Narrative Sorcery: Coherent Storytelling in an Open World,” presented at the GDC 2017, 2017.

[31] G. Smith *et al.*, “Situating Quests: Design Patterns for Quest and Level Design in Role-Playing Games,” in *Interactive Storytelling*, vol. 7069, M. Si, D. Thue, E. André, J. C. Lester, J. Tanenbaum, and V. Zammitto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 326–329.

[32] T. Hromada, M. Černý, M. Bída, and C. Brom, “Generating Side Quests from Building Blocks,” in *Interactive Storytelling*, vol. 9445, H. Schoenau-Fog, L. E. Bruni, S. Louchart, and S. Baceviciute, Eds. Cham: Springer International Publishing, 2015, pp. 235–242.

[33] B. Kybartas and C. Verbrugge, “Analysis of ReGEN as a Graph-Rewriting System for Quest Generation,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 2, pp. 228–242, Jun. 2014.