DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Games Engineering

Information Flow in Distributed Multi-Agent Systems as a Game Mechanic for Immersive Story Worlds

Jakob Raith

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Games Engineering

Information Flow in Distributed Multi-Agent Systems as a Game Mechanic for Immersive Story Worlds

Informationsfluss in verteilten Multiagentensystemen als Spielmechanik für immersive Geschichtswelten

Author:	Jakob Raith
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	Daniel Dyrda, M.Sc.
Submission Date:	15.10.2021

I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.10.2021

JAKOB RAITH

Abstract

The demand for high-quality video games is ever increasing. This entails every aspect of video game creation, such as graphic fidelity, sound design, gameplay innovation, or novel game design approaches. However, the ambition to tell truly interactive and dynamic stories is also a significant factor contributing to this trend. This thesis aims to create a novel game mechanic that supports narrative gameplay. The presented game mechanic is based on exchanging information between the player and NPCs and among NPCs themselves. By modeling the NPCs as intelligent agents in a multi-agent system, techniques from this field are used to give the agents an autonomous behavior as they move through a 3D environment. A classification for different types of gameplay focusing on dynamic gameplay is created, and the developed prototype is compared to other current games in this classification. The resulting system creates a playground for varied narrative scenarios that are not previously defined by a game designer but can be explored by the player and are supported by the game systems.

Acknowledgements

I want to thank several individuals and institutions for helping me in completing this thesis. First and foremost, I want to thank Prof. Gudrun Klinker, Ph.D., and the Department of Informatics of the Technical University of Munich for enabling me to do this thesis. I also want to thank Daniel Dyrda for supervising my research and helping me with research pointers, valuable insights, and personal support. Furthermore, I want to thank Dominic Ford, a researcher from Denmark, who also provided relevant resources and support for my research. Finally, I would like to thank my family and friends for lots of support, patience, and kind words during this work-intensive time.

Contents

1 Introduction

Video game players today ask for narrative-focused games more and more often. A report from WePC.com in which 73.55% of people who play video games prefer single-player games over multi-player games indicates this fact [99]. This is, however, not to say that multi-player games can not focus on the story. It is common, though, for single-player games to have a stronger focus on narrative. WePC.com further notes that gamers claim to play more video games because of the COVID-19 pandemic. Between March 23 and June 3rd, 2020, the number of players increased by 46% in the United States, 41% in France, 28% in the United Kingdom, and 23% in Germany. [99]
The rise in interest in single-player games is also backed by Sony Interactive Entertainment. With 46% of the console market, Sony is the leader of a $58.6 billion industry segment. [21] A supposedly leaked document mentioned in a Vice.com article reports that the Sony console's internal tracking tools found that PlayStation users spend more time offline than online regularly [52]. Finally, in an interview from March 2020, the head of PlayStation's Worldwide Studios, Hermen Hulst, says that Sony is "very committed to quality exclusives. And to strong narrative-driven, single-player games." [89] PlayStation Studios consists of 14 individual studios that focus on narrative-driven single-player games [92]. All these commitments of the console industry leader to storytelling clearly indicate that the demand for games with a narrative focus is present in the video game landscape.
Modern narrative games feature stories that rival movies in terms of production value. In May 2018, Eidos Montreal boss David Anfossi said that the game *Shadow of the Tomb Raider* [29] cost $75 - $100 million just for production [27]. These more expensive games usually feature a very cinematic and action-oriented storyline with well-known voice and performance actors. They are also usually quite linear in their structure and storytelling. Developers want to ensure that players actually see and experience the expensive set-pieces in the stories, which explains this more linear approach to storytelling.
However, would interactive media like video games not profit from the potential of actually interactive storytelling? Players often wish for games to give them many options in their interactions. "I want to be able to do whatever I want" is a request that comes up frequently when talking about open-world games. Acclaimed game designer Sid Meier in his 2012 talk at the Game Developer Conference, describes games as a "series of interesting decisions". He explains that by limiting and carefully crafting the actions a player can take, the gameplay becomes more compelling [27].
So the argument can be made that "being able to do whatever I want" could actually be detrimental to the game experience. This argument can be extended into the story. Authored stories often follow the rules of their respective culture's storytelling traditions. We perceive stories that deal with culturally relatable topics and follow familiar structural rules more favorably. [16]
However, would players be able to make interesting narrative decisions naturally, given the option by a game that does not restrict interactions to an authored story? Does a video game story require an author at all? Can we develop systems that take on that role? Could a game mechanic provide all necessary elements to satisfy narrative require-

ments? This thesis aims to explore not only what those requirements are and how they contribute to game stories, but also introduces a framework for game mechanics that allow the utilization of world state information for narrative purposes.
A story can be abstracted as numerous pieces of world information that are exposed to the person experiencing the story, in this case, the player. If we have a story that goes like "The knight slays the dragon." We know that the world consists of a knight, a dragon, and the act of slaying the former. These pieces of information about the actors and acts in the world are then framed into a narrative structure - the story.
In this thesis, I developed a game mechanic based on modeling this kind of information into objects that exist in the game world. They are created by interacting with the world and are perceivable by agents in the system next to the player. By allowing the non-player agents to react to newly learned information, the model creates an interplay among all agents in the system. The resulting game mechanic then is based on retrieving relevant information about characters or other objects in the game world and introducing new information as the player. This mechanic could even be used to create factually wrong information and thus allow the concept of lying to agents.
For the prototype game that was also developed as part of this thesis, I chose the following setup to showcase the game mechanic. The player controls a knight in a freely traversable 3D environment that represents a medieval kingdom. There are several key locations like a castle, a village, or a farm that are inhabited by other *non-player characters* (NPC) that represent the agents of the multi-agent system. The NPCs move around the kingdom and execute different tasks, which bring them into contact with each other. When they interact, they will also exchange information and thus learn new things about the game state. The player can also interact with characters and items and learn pieces of information. They can also engage in a conversation with NPCs and ask for or give out specific information. By adding quests to the game, a narrative framing exists that the player can use to motivate their actions.
Elevating the exchange of information in multi-agent systems to a core game mechanic and allowing players to exploit the information flow is a novel system that not many games support today. It is also a step towards creating an actual interactive story because it allows for genuinely influencing the game world but still enabling developers to define rules for how agents and the environment reacts and thus enforcing a narrative structure.

2 Theory

This chapter presents the various fields of study that were encountered or utilized while developing the game mechanic and its implementing prototype game. This includes narratology, game and quest theory, information theory, multi-agent systems, consensus protocols, emotion engines, and inference engines.

2.1 What is a story?

There exist a multitude of definitions of what a story is. Rayfield argues that a story is a narrative item that exists throughout all cultures. He concludes that there exists a universal concept of a particular structure that listeners will recognize as a story. He limits this structure by the degree of complexity and argues that listeners will only recognize the structure as a story within certain minimal and maximal bounds of complexity. These bounds would then be the same across all cultures. [78] Scheub takes another approach and sees story more as "a means whereby people come to terms with their lives, their past; it is a way of understanding their relationship within the context of their traditions. It is a means of accessing and valuing history: in the end story *is* history." [84] Lastly, the Oxford Dictionary of Literary Terms more formally describes a story as a set of events that are selected and arranged in a specific order and told by a narrator. The specific order of the events is called the plot. [5]
One thing that most definitions have in common, however, is that stories are an essential tool for humans. Stories help us to interpret and process information and experiences. They enrich subjectively perceived facts and form them into each person's individual truth. This is the *"meaning"* of a story. This "meaning-making" is also part of the psychological process in self-identification and the creation of memories [32]. Stories are furthermore a crucial factor in human communication, used as parables and examples to illustrate points. Storytelling was one of the earliest forms of entertainment.
When looking into definitions of what a story *is*, the matter quite quickly goes into the area of what a story *does*. I already mentioned how it is a vital tool for the human psyche, but there are some more concrete functions that stories fulfill.

2.1.1 Functions of Storytelling

The motifs and contents of stories and the modes of narration are highly culturally individual aspects. The functions these elements serve, though, can be found across all cultures. One example is to make one narration more understandable by putting it into relation with another story. This is commonly referred to as a metaphor. Now, these relational stories do not have to be imaginary per se. They can be a retelling of events that have actually transpired and help underline the narrator's point. On the far hand of abstracting stories to make a point is the cautious use of words to find an objective true transpiring of events. This is what happens in courtrooms. It is a retelling of events, but

the order and selection are so careful, so meticulous, that an actual fair "true" story might be revealed. [81]
As mentioned above, stories and storytelling are used to share and interpret experiences. The human brain is evolutionarily predisposed to process, store and recall memories in the form of stories [102]. Humans think in narrative structures and mostly remember facts in the form of a story. Facts are smaller versions linked to a larger story, which supports analytical thinking. [14]
This makes storytelling such an excellent tool for teaching as well. There is research about how storytelling is a meaningful teaching method that can be applied in education to encourage the development of caring, empathy, compassion, and a deeper cultural understanding. [20]
It is not event solely the listening person who is learning from a story. Often the discovery of a personal meaning of a story is only made visible when telling a story. Thus also the storyteller can learn something new. [25]
This is applied in therapeutic storytelling, where through retelling experiences in story form, the storyteller attempts to understand their own thoughts and situation better. This can be supported by questions from a therapist who carefully steers the storyteller through their narration to pinpoint insights. [58]
There are countless situations where we encounter storytelling precisely because we want to share experiences and emotions. Stories are used to inspire and motivate, manage conflicts, for marketing, or for political practice [49]. These are all situations where the intent is separate from the story itself. Where we turn to storytelling because the human brain can process them so effectively. Another aspect is, however, when we tell stories for the story's sake.

2.1.2 The Appeal of Stories

I have now established that there is a difference between the intention of storytelling itself and utilizing it for another purpose. The difference is that we do not enter a courtroom to tell or listen to a story. We do so to find the truth. The past has shown us that the careful recounting of events combined with precise inquiry has proven an excellent way to do so. So when we do not tell stories as a means of achieving another intention, we also share them just because they are stories. For this, we can take on both positions, that of the narrator or the listener. Again, a multitude of situations presents itself for these applications. We listen to bedtime stories that our elders tell us. We tell a funny anecdote to our friends on a night out. Humans have created whole industries around the consumption of stories. We read books to immerse ourselves into stories, we watch movies or shows, and we play narrative video games. The fact that so much of our time is willfully spent listening to, watching, or interacting with stories, must mean that there is something worthwhile there. The mere consumption of narration seems to be satisfying in itself.
When we hear stories, the brain releases the hormone Oxytocin. This hormone heightens feelings of trust, empathy, and compassion. It positively influences social behavior and helps us to feel more connected to others. [39] Humans are inherently social beings. We want to connect to others around us. We try to create situations and use known cultural signals to connect on a non-verbal level. Humans mimic body language, laugh more in a social situation than alone or use physical contact to communicate. [36]
When we hear a story, the brain is enabled to form connections to the people who listen to the story with us, to the narrator, and to the characters in the story. Communication is

a shared activity resulting in a transfer of information across brains. Research shows that during successful communication, the brains of both the speaker and the listener shows common, temporally linked response activities. When we hear a story, our brain mirrors activities in the sensory center of the storyteller. [95] This attempt to sync brain activity is a profoundly social connection. It also means that when we hear an enjoyable story, the brain behaves as if we would experience it ourselves.

2.1.3 Stories in Media

The consumption of stories has led to the creation of huge industries that focus entirely on creating and delivering stories to customers around the world through different kinds of media. That term comes from the Latin word *medium* for "middle", which again stems from the ancient Greek word *méson* for "the middle" or "the public". Today we use the term media as a word for "the means of communication". [44] In the context of storytelling, this could be oral through a present storyteller, audio through an audiobook, visual through a book or e-book, audiovisual through theater plays, movies or series, or interactive through games and other interactive media. Oftentimes, when we talk about "the media" today, we refer to an industrialized consumption of content or news.
The economic factor of these storytelling media cannot be underestimated. The global Entertainment & Media Outlook analysis from PriceWaterhouseCoopers reports over US$ 40bn for global box office revenue in cinemas for 2019. Cinema revenue, of course, took a nosedive because of the COVID-19 pandemic in 2020 and 2021 but is expected to recover and grow by 2023. Video games, however, have been steadily growing to roughly US$ 62bn for traditional games revenue and almost US$ 92bn for social or casual gaming revenue in 2021. These numbers are also expected to grow in the coming years. [77]
Storytelling, of course, works differently and has different requirements in each of these mediums. Books, for example, allow for a seamless switch of inward and outward perspectives of characters or thought processes. Movies as an audiovisual medium are more restricted to an outward perspective and have to use different storytelling techniques to transport the inner feelings of characters. There is also a structural difference to all these forms of media since there are conventions for runtime for movies or an episode of a show. [83]
The requirements for video game stories stem significantly from the fact that there is a distinction between the human player in front of the screen, which controls a player character, and this character who inhabits and acts in the game world. In the story that we experience in the game, the plot points that we traverse were aptly defined with the term of narrative by Abbot. Similarly to Baldick in the Oxford Dictionary of Literary Terms, he says that narrative is the representation of an event or an action or a series of events or actions. He further argues that without action, only description or exposition remains [2]. This leads to the idea that narrative can only exist when there is a transition from one state into another. This transition of states is then an event or an action. It fits well with how Aristotle described a story as a whole that "[...] has a beginning and middle and end." [4]. This idea also implies the existence of at least three states in a story, the beginning, the middle, and the end, in addition to the transitions in between. These transitions would be, according to Abbot, the narrative.

2.1.4 Expectation & Feedback

Players have certainly learned expectations when controlling a player character in a narrative game. They expect their gameplay actions to represent the motives of the character on the screen faithfully. When I talk about the player character, I refer to the virtual character in the game world as a function of the human player who controls them on-screen. It is the character we follow in the story. In their work about Character and Conflict, Lankoski and Heliö argue that a narrative state transition, an action, is the most crucial feature from the player's point of view [56]. They also argue that character-driven action is very similar to character-driven writing, a similarity that is especially important for story-driven games. They say, "[...] instead of writing a story, characters and their needs are designed so that there will be enough action and conflict in the game to make it playable and interesting." [56] This makes motivation or needs so essential for directing action and keeping a game interesting.
This idea of actions that make a game playable and enjoyable is well explained with the definition of agency by Wardrip-Fruin et al. They define agency as a phenomenon involving both player and game. It "occurs when the actions players desire are among those they can take (and vice versa) as supported by an underlying computational model." [71] They shift the focus of a definition of agency toward a question of how games can evoke desires that can be satisfied by the game systems or how said systems can be created, towards how to manage player expectations and early on explain the game systems so that player expectations and thus player agency can adapt. This finally brings me to the term of the avatar. Klevjer agrees that avatars are "little more than a cursor" to facilitate player agency in the game. They are tools to fulfill the actions that the player wants to perform [53]. So the term avatar stands on the opposite side of the term player character. One is the tool used by the player to create meaningful actions, and the other is a character in a narrative with goals and motivations. If the game manages to merge the motivations of the player and the player character, then the game succeeds in creating agency.
This makes agency one of the primary requirements for a video game story. Callele et al. call the game designers' intent or the target emotional state and the means by which the game designer expects the production team to induce that emotional state in the player *the emotional requirement*. This requirement needs game developers and storytellers to align their intentions carefully. They need to take cultural, spatial, and relational aspects of gameplay events and symbols into account to ensure that their intended emotion is transferred to the player. [11]
Another requirement for game stories is one of more concrete communication. Games convey meaning different from other forms of media. Every game brings with it a specific set of semantics, the game rules that need to be correctly communicated to the player in order for them to be able to play it. A player who understands the game's rules knows the available actions they can take. This allows them to adjust their expectations to their character and the story. If the story presents the player character as a legendary warrior capable of defeating any enemy, but the game gives me no option to engage in combat, the player will possibly not understand how the reputation of the character came to be in the first place. Likewise, a pacifistic character with an arsenal at their disposal will create a similar conflict of intentions between the game and the player. Rule conventions also sometimes dictate feedback that the player expects. An example would be if the player triggers the input for a *move right* action, they expect the character to move to the right on the screen. This creates an interplay between expectations and feedback that comprises

both gameplay and narrative. Symbols come with certain expectations for actions to be available to the player, and events triggered by actions come with certain expectations of feedback to be received from the game.

2.1.5 The optimal Experience

A synchronization between expectation and feedback is also something Csikszentmihalyi describes when he talks about the *optimal experience*. He describes incidents when a sense of exhilaration and of deep enjoyment is felt. These moments are often highly active and not receptive or relaxing times. They occur especially when we engage in a voluntary endeavor to do something difficult or worthwhile. Optimal experiences result when there is order in the consciousness. This happens when we are focused on a realistic set of goals with our skills matching the opportunities for action. It can be easily seen how this translates very well to gameplay in video games. Goals allow people to concentrate attention on the task at hand, with other things fading into the background. The key element of an optimal experience is that it is an end in itself. It may be taken on for other reasons initially, but the activity soon will become intrinsically rewarding. It has to be autotelic. The term is formed from the Greek words for *self* and *goal* respectively, and means the activity has an end or purpose in itself. [18]
I have established earlier that experiencing stories is, in fact, an autotelic activity. With the science of optimal experience, we can assume that a game mechanic that could satisfy narrative requirements would indeed create a gratifying gameplay experience.

2.1.6 Quests

Narrative-focused games often feature a particular concept for compartmentalizing their stories - the *quest*. The word has Latin origin in the word *quaesta* for inquire or search. In literary it has long been known the describe the difficult, often symbolic, or allegoric journey towards a goal. It plays a central role in Joseph Campbell's *"Hero's Journey"*, where he says, "*[...]the hero sets forth from the world of common day into a land of adventures, tests, and magical rewards. Most times in a quest, the knight in shining armor wins the heart of a beautiful maiden/ princess*" [12]. Video games also have been using the term for decades for a isolated concrete set of goals that marks a progression. A quest is a task, a mission. It has a set of goals that need to be completed in order for the quest to be finished.
Aarseth defines three basic quest types that can be combined and rearranged to create complex structures:

- **Place-oriented**: The quest requires the player to move from a starting position A to a target position B. Typically, the player will encounter obstacles along the way, but the basic premise remains.
- **Time-oriented**: These quests add a time-related aspect to the goal. For example: *"Survive for X seconds."* or *"Find three apples in Y seconds."*
- **Objective-oriented**: This includes a concrete result to the goal requirement. For example, *"Get item I from character J."* A specific condition needs to be met for the goal to be finished.

The basic types can be combined, nested, serialized, or parallelized in order to create highly complex quest worlds. Common types are the serial arrangement of quests into a linear corridor, a semi-open hub, and an open landscape. [1]

Open quest-landscapes are often featured in open-world role-playing games like *The Elder Scrolls V: Skyrim* [7]. Interestingly enough, the hub structure mimics the literary hero's journey quite well. It has a distinct "home base" from where the hero ventures out into the unknown and dangerous lands to face obstacles, only to return to his home a changed person. This is a structure that has been well researched in many old texts, such as the myths and tales centered around King Arthur. Spearing describes this in his work on Sir Gawain and the Green Knight. A knight in search of honor leaves the round table and the comfort of Camelot to seek his fortune. He faces dangers and characters that change him and eventually returns home. He now understands this place of familiarity differently, not because *it* has changed, but *he* has. [93]
It has been on purpose that I wrote of the game and the narrative as two distinct, even if interlinked, entities. I agree with Aarseth that what may resemble narrative structures is actually spatial (go from A to B) structures. Games that might appear the most story-like, are in fact, often reliant on place-oriented quests and very spatially constrained. The challenge for game developers, as he puts it, is to "*[...] move beyond the constraints of unicursal corridors or multicursal hub structures while keeping the player's attention on a storyline. And that is no easy task. But perhaps presenting an interesting landscape with challenging quests is enough?*" This is precisely what the prototype of this thesis attempts to do. [1]

2.2 Game Mechanics

In the previous sections, I talked at length about the different requirements for a story in video games and how the narrative relates to the gameplay. In this part, I will discuss the topic of game mechanics in general. To talk about a story *as* the game mechanic through the exchange of information, I need to provide sufficient definitions in both areas.
One robust definition was provided by Cook in 2006. He describes game mechanics as "*rule based systems / simulations that facilitate and encourage a user to explore and learn the properties of their possibility space through the use of feedback mechanisms.*" At the center of this definition are feedback loops for learning new information. The definition is well illustrated as a diagram (Figure 2.1). [15] For my project, I chose to follow Sicart, who describes another suitable formal definition of game mechanics. It stems from object-oriented programming while still keeping many ludological aspects such as the figure of players or agents as fundamental parts to understand how games are designed and played. He does so by purposely distinguishing between game mechanics and game rules. The former are the options of interactions available to an agent in the game. The latter are the restrictions under which those options are made available and how they change the game state. Sicart arrives at the definition of game mechanics as "*methods invoked by agents, designed for interaction with the game state.*" The first part of this definition is rooted in the paradigm of object-oriented programming. This appropriation of terminology is helpful because it provides a formalistic perspective to actions performed in information systems like games. This can even lead to the application of modeling languages like UML (Unified Modeling Language) to the description of game systems. [90]
In object-oriented programming, a method is the actions or behaviors available to a class. Methods are the functions an object has for calculating data or interacting with other systems or objects. [97]
Following this logic, a game mechanic is an action invoked by an agent to interact with the game world, as constrained by the game rules. *Super Mario 64* [69] usually allows the player to jump. If Mario is standing on quicksand, however, he will slowly sink into the

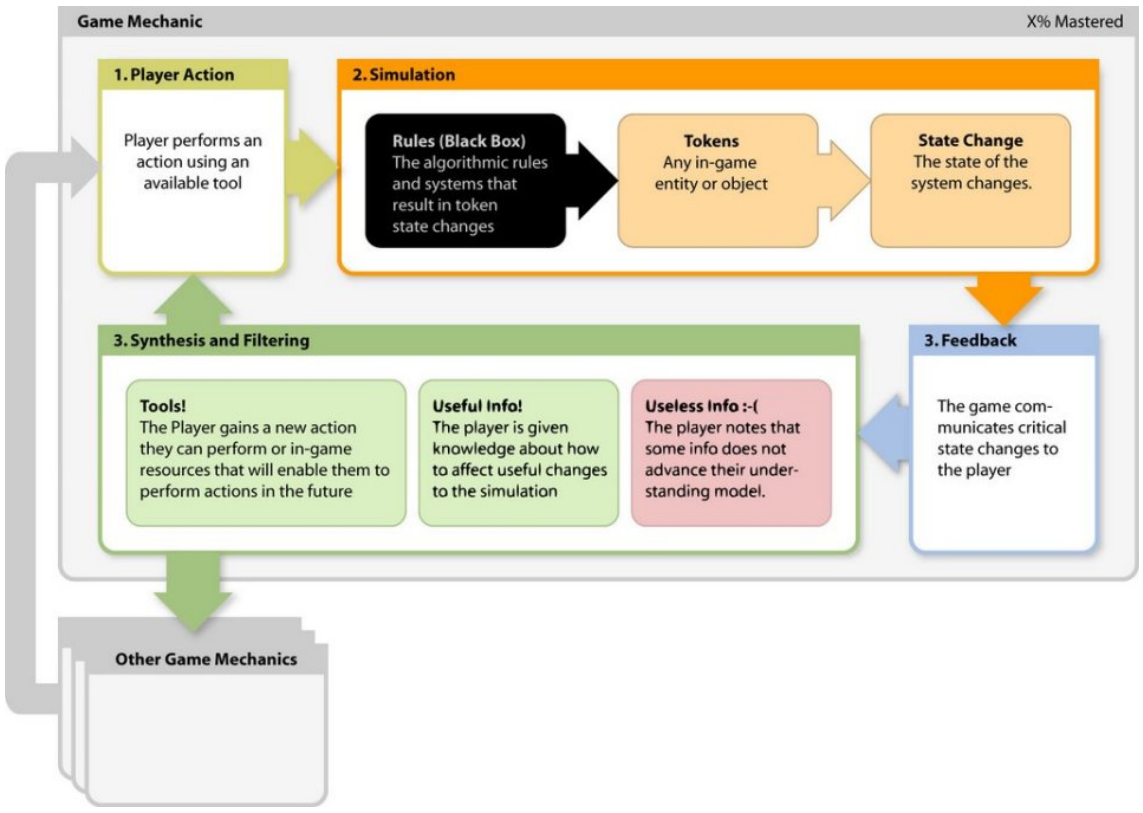

Game Mechanic
X% Mastered
1. Player Action
Player performs an action using an available tool
2. Simulation
Rules (Black Box) The algorithmic rules and systems that result in token state changes
Tokens Any in-game entity or object
State Change The state of the system changes.
3. Feedback
The game communicates critical state changes to the player
3. Synthesis and Filtering
Tools! The Player gains a new action they can perform or in-game resources that will enable them to perform actions in the future
Useful Info! The player is given knowledge about how to affect useful changes to the simulation
Useless Info :-(The player notes that some info does not advance their understanding model.
Other Game Mechanics

Figure 2.1: Cook's definition of game mechanics [15]

ground, and jumping repeatedly will only bring him back to the surface. That means a mechanic is limited by the rules that apply to the game world, such as physics simulations. Mechanics can also be limited by rules that are applied to mechanics in specific game state contexts, like being only able to attack when a weapon is equipped.
The object-oriented approach of invoking methods for interactions decouples the interaction from actual player input. On this systemic level, Sicart is able to talk just about agents that are interacting with the game world and not necessarily player interactions. Game mechanics can be invoked by any agent in the game, be that human or part of the computer system. Agents using artificial intelligence also have a number of methods available to interact with the game world. How actual player interactions are then mapped to input devices can be viewed separately.
The second part of the definition says that game mechanics are methods "*designed for interaction with the game*". These interactions modify the game state. Game developers design basic mechanics that enable the player to overcome obstacles in the game. Challenges here describe gameplay situations in which the player has to invest an effort to progress further. All challenges are matched with a mechanic: by jumping, Mario can reach a higher-up area. By striking with an equipped weapon, a character can damage an enemy. The fact that games are structured as systems with mechanics, rules, and challenges is understood as the essential grammar of video games. [90]
I have touched upon the fact earlier that there is more to the act of playing a game than just interacting with mechanics constrained by rules. In the act of playing, players will appropriate agency within the game world and behave in unpredictable ways. As mentioned before, it is one thing that a designer intends for the player to feel, and another, how players actually interact with the game world. The formal, analytical understanding

of mechanics only allows us to design and predict courses of interaction but not to determine how the game will always be played or what the outcome of that experience will be. Designers can build rules and challenges with specific emotions in mind that they want to invoke in the player and then create matching mechanics that allow overcoming the challenges and thus create other resulting emotions.
This systemic approach makes two methods of analyzing game mechanics possible: For one, it allows to systemically analyze the structure of games in terms of actions available to agents to overcome challenges. The second perspective is the analysis of how actions are mapped onto input devices and how mechanics can be used to create specific emotional experiences in players. Sicart does, however, point towards gray areas in his definition. Perhaps the most significant is the categorical distinction between rules and mechanics. It is part of ongoing scientific discourse if mechanics are, in fact, a subset of rules. He argues, however, that rules are normative while mechanics are performative and that, therefore, this ontological distinction can be beneficial for the analysis of computer games. Game studies history shows that there is no dominant definition of key concepts like rules or mechanics. It is, in fact, the core message of Sicart's work that it is possible and valuable to understand game mechanics as different from game rules, and in that understanding, it can more clearly be described how games can be designed to affect players in new and innovative ways. [90]

2.2.1 Emergence Games

The game mechanic that is developed as part of this thesis should allow players, as stated, to introduce, exchange, and retrieve pieces of information from a system of agents for the purpose of very individual narrative paths. This *very individual* aspect of the mechanic invites the concept of emergence into the design process. Juul wrote about emergence games as opposed to games of progression.
Emergence happens when the whole is more than the sum of its parts when the activities of the parts do not simply add up to the activity of the whole [45]. Juul describes emergence games as those that have a small number of rules which combine to complex and varied results. It is in those games, where players tend to create strategies for dealing with the encountered results. Typical examples for games of emergence are card and board games, most action games like fighting games, and all strategy games. Replayability is naturally high in emergence games, as are tournaments and existing strategy guides. Adventure games then introduced progression into video games. The player is confronted with a series of challenges that they have to overcome in a predefined way and order. Games of progression can be controlled by game designers much easier and more tightly. Historically this is also where most narrative games find themselves since the more linear and controlled sequence of events lends itself to traditional western storytelling techniques. For that reason, progression games more likely have so-called "walkthroughs"; guides specifying all the ordered actions and steps needed in order to complete the game. [50]
Further, both of the concepts are not mutually exclusive. Games of emergence can feature progression structures and progression games can contain emergent mechanics. Juul defines several types of emergence:

- **Rule interaction**: simply two or more rules yielding a new result when satisfied at the same time. An example would be that trees catch fire when struck with a lit torch in *The Legend of Zelda: Breath of the Wild* [70].

- **Combination**: This type of emergence stems from the different potential game sessions that are created through the combination of different rules — for example, the procedurally generated 2D levels in *Spelunky* [66].
- **Emergent strategies**: These are what Juul calls the actually emergent properties since they are not immediately derivable from the basic game rules. There are countless examples for this type, such as all game strategies or the need for cooperation with other players.

Emergence does, however, not exclude all kinds of direction from designers. Games of emergence lie somewhere in between complete open games where the player is free to do what they please and closed up games where everything that can happen is predetermined. [50]
This makes the concept of emergence so attractive for this thesis since it would allow some form of narrative structure to remain in a system where the simple rules enable the player to come up with new and exciting solutions.

2.2.2 Multiplicative Game Design

Staying on the broader topic of emergence, another related concept is helpful for designing game mechanics. *Multiplicative gameplay* was used by the developers of *The Legend of Zelda: Breath of the Wild* [70] to describe their design approach of what they called "*rediscovering the essence and breaking conventions*". Early in the development process, the game designers realized that they wanted to make the player feel a new sense of adventure again and again while having freedom of where to go and what to do. They realized, however, that previous *Zelda* games had the following conventions that would keep them from that:

- a predetermined sequence of events
- non-traversable walls and barriers
- a predetermined experience
- widely available answers for puzzles online

Interestingly these points sound precisely like what Juul describes as games of progression while the developers of Breath of the Wild wanted to create an emergence game. They did so by carefully examining the elements that were progressive in nature, removing them, and replacing them with systems that have emergent properties. Insurmountable cliffs, for example, were made climbable, and a stamina system for climbing was introduced. This was expanded to the whole 3D world, which immediately made the entire landscape traversable, given enough stamina. They applied interlinking systems to all their game objects which creates an immense variety of what Juul calls combination type emergence [50]. By embedding these systems into a robust physics engine to allow for varied but expected behavior, the developers were able to create satisfactory gameplay feedback for players, as discussed in section 2.1.4. Exactly this interlinking of gameplay systems is what the developers call multiplicative gameplay. In *Breath of the Wild*, they could even leverage the variety of interaction options to design the small shrine puzzles in the game. Because of the emergent nature of the game, the shrines often do not have a single correct solution but many that allow the players to explore and play around with mechanics. [37]

2.3 Concretizing Information

In order to create a game mechanic based on game world state information, this information needs to be modeled in a way that can be interpreted by game rules. Information itself is quite an abstract thing until saved in some form of permanent medium like writing it down. It seems ironic that a general definition of information becomes ambiguous quickly. Etymologically the word comes from the Latin verb *īnfōrmāre*, which translates to "*to give form to the mind*". Information has not only different meanings in different contexts, but even when just talking about information as something that is "informative", it becomes difficult to pin down. It can be the process of being informed, the thing one is being informed about, or a tangible thing that informs and "is informative". [9]
There has always been a philosophical aspect to discussing information. A data-based definition of information seems most intuitive despite the fact that the nature of data is not well understood philosophically either. Nevertheless, data is a less substantial concept and makes it easier to grasp.
The General Definition of Information by Floridi states the following: GDI) σ is an

instance of information, understood as semantic content, if and only if:

GDI.1)	σ consists of n data (d), for n $\geq$ 1;
GDI.2)	the data are well-formed (wfd);
GDI.3)	the data are meaningful (wfd $= \delta$);

This tells us a number of things. Firstly, information is made from data (GDI.1). Secondly, that data needs to be "well-formed", meaning it needs to be structured or bundled in the correct syntax, i.e., the rules of correctness for the analyzing system (GDI.2 and the first branch in Figure 2.2). Finally, GDI.3) tells us that the data needs to be meaningful according to the semantics of the system (the second branch of Figure 2.2). Both syntax and semantic are not necessarily linguistic constraints, but whatever those rules are for the governing system (e.g., a programming language or movie script format). Following the GDI information cannot be "dataless" since even the lack of data or unavailability of data is data in itself. [33]
In Figure 2.2, we can see a clear distinction between the different qualities of information, including factual properties. The leave nodes in the diagram denote the following types of information:

- **Environmental**: Observed data that can be natural (e.g., rings in a tree trunk to estimate age) or engineered (e.g., the temperature from a thermostat).
- **Instructional**: Semantic data that conveys the need for a specific action (e.g., order, instructions, stipulations, invitations).
- **Misinformation**: Factual, wrong data, and the source *is not* aware of its nature.
- **Disinformation**: Factual, incorrect data and the source *is* aware of its nature.
- **Knowledge**: Factual, accurate information.

It seems clear how fast an attempt to define and classify information generally becomes a matter of philosophy. [34]
The focus on semantics differentiates this more philosophical concept of information significantly from information theory. The science concerned with quantification, storage, and communication of digital information sees information entirely technical.

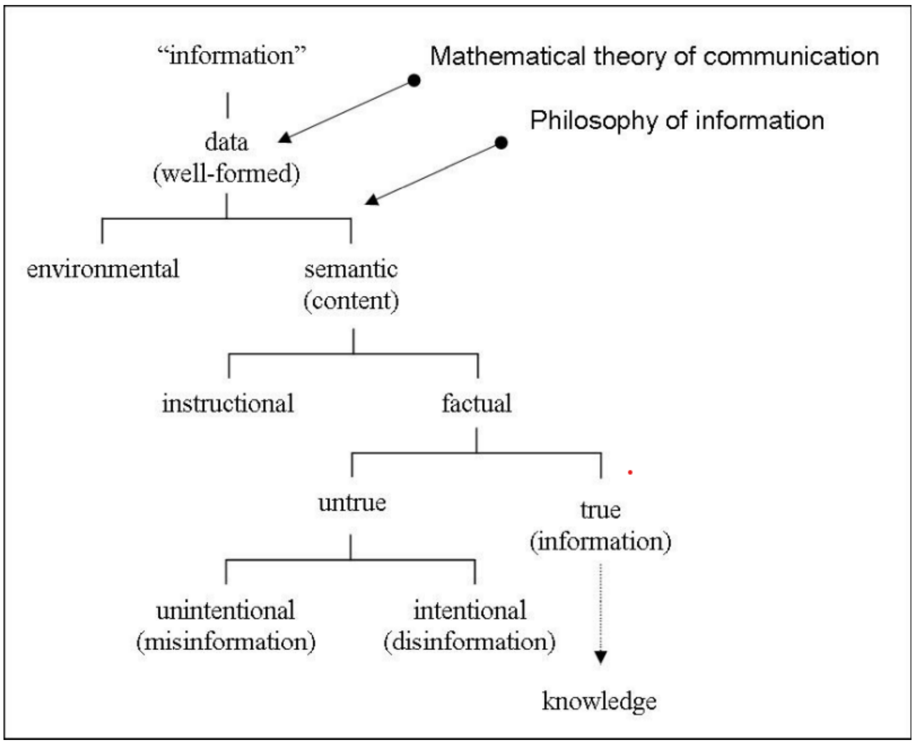


Figure 2.2: A data-based classification of information [34]

The definition relates to the resolution of uncertainty. It understands information as that which reduces uncertainty. Now probability mathematics allows quantifying uncertainty, and that means information can equally be quantified. For sufficiently communicating information, information theory introduces the *bit* as the unit of information. A bit is in a sense "*that which halves uncertainty*" since a bit can only have two states, 0 or 1. [86]
The information value of a given symbol is the uncertainty that the symbol is transmitted. That means the rarer it is for a symbol to be transmitted, the more information value it has. The logarithmic function is used to express the information value I of symbol x in a bit.

$$I(x) = -\log_2 p(x) \tag{2.1}$$

Where $p(X)$ is the probability of the symbol x being transmitted, then the average information value of a transmitter X is called entropy.

$$H = \sum_{x \in X} p(x)I(x) = -\sum_{x \in X} p(x)\log_2 p(x) \tag{2.2}$$

All this, of course, relates in practice to the transmission of information but still establishes a sense of measuring "informativity". [87] Evaluating information based on its rarity can, of course, be useful for a game mechanic based on exchanging information.

2.3.1 Communication

The prototype game created as part of this thesis aims to emulate human communication by giving agents the ability to exchange information. Exchanging information is,

of course, nothing else than communication. In order to simulate this, I looked into human communication. As being with the capacity for social interaction, the development of speech and the formation of languages is unique in humans [59]. Communication is another concept that has been often researched and differently defined by scholars [60]. However, there exists some consensus that the first evolutionary usage of spoken language was the representation of one's thoughts. This was followed by the function of exchanging ideas [82]. Of the many available ones, I will choose the definition given by Adler et al. that "*communication is the process of creating meaning through symbolic interaction*" [3]. It is described as a symbolic, relational process. Symbolic, as communication uses agreed-upon symbols to codify meaning, relational as communication needs adaption and coordination of the participating parties, and it is a process as it is not over after an isolated utterance or conversation. Sometimes the processing of received communication changes the thinking of the receiver over hours, days, or years. [3]
However, human communication is imprecise. As humans, we are to a degree at the will of our emotions, and this means they also dictate the way we communicate with others. There are many reasons that can make human communication ineffective:

- **Words are imprecise**: the languages we use for communication consists, of course, of words, and sometimes the words fail to describe complex subject matters universally yet accurately.
- **Coded language**: Humans always communicate somewhere in-between, wanting to be known and wanting to stay hidden. We use "codes" both involuntarily or not to make our messages more indirect or ambivalent.
- **The emotional factor**: As mentioned, emotions highly influence the way we communicate and how we receive what is communicated to us.
- **Attention and distractions**: When distracted or focusing on something else, it only makes sense that we either are unable to communicate effectively or receive everything correctly.
- **Reception filters**: Society conditions us to filter out some information channels to protect us from being overwhelmed. Also, conditioning through teaching or experience will create certain filters when receiving information or even may filter how information is interpreted depending on who the sender is.
- **Transmission interference**: Simple physical conditions can also contribute to an error in communications. There could, for instance, be a loud background noise that drowns out a verbal utterance or a stain on a book page covering an important written word.

These problems are not only restricted to human verbal communication but can occur, in varying degrees, in all kinds of communication. [67]
In addition to these universal human conditions, society is not a homogeneous group. Cultural differences, the purpose of communication, or the number of conversation participants also greatly influence the process [13]. These factors of imprecision of human communication all contribute to a difference in the intended information from the sender and the information that is understood by the receiving party. I collect and describe this difference under the umbrella term "*Information Mutation*".

2.3.2 Information Mutation

I established that the relational aspect of human communication introduces a number of problems that need to be addressed and accounted for in everyday life. However, not only is it challenging to preserve the integrity of information while it is sent, transmitted, and received, retaining information as the receiver also comes with its own set of hindrances.
The part of the human brain responsible for storing information is, of course, the memory. In the following paragraphs, I will use the terms information and memory more or less interchangeably for better understanding. With memory, I mean the encoded information in the brain. We base our actions and decisions on experience that is stored in the memory in the form of memories [88]. The existence of memories is a necessity for the development of language, relationships, or even personal identity [30].
Disremembering or forgetting is something everyone experiences. It is the apparent loss or modification of information that was already encoded and stored in the short or long-term memory. Even when it is not due to an illness or defect of the brain itself, humans forget things. Forgetting can be both spontaneous or a gradual process after which old memories are unable to be retrieved from the memory. Studies show that retention of information improves by increasing rehearsal. Rehearsal helps to transfer information to long-term memory. [98]
It seems to be a natural part of the brain's function, but there exist multiple theories on why that is.

Decay Theory

The Decay theory proposes that memory fades as a function of the passage of time. What that means is that information is less available for retrieval as time passes and memory strength decreases. The new acquisition of information creates a neurochemical "memory-trace". As time passes, this trace slowly decays. Scientists suspect that active rehearsal or repetition helps counteract this effect [72]. Although it is assumed that neurons in the brain slowly die off with increasing age, there are older memories that can be stronger than younger ones. The Decay Theory applies, therefore, especially to short-term memory. That means older memories in the long-term memory are often more resilient against shock, trauma, or physical attacks on the brain. The passage of time is also probably not the sole reason for forgetting. [6]

Interference Theory

Interference occurs when learning new information. The Theory proposes that memories encoded in the long-term memory are being forgotten and cannot be retrieved into the short-term memory. This would be because a memory in one interferes with the memory in the other. There is an immense amount of encoded memory in long-term memory. The challenge of remembering is to retrieve a single specific memory from long-term memory and transfer it into short-term memory for processing. There are two types of interference effects, proactive and retroactive interference. [28]
Proactive interference is the inhibition of the retrieval of new memories caused by old memories. Of the two effects, proactive interference is the rarer one and the less severe one. The hypothesis exists that without proactive interference, it would be impossible to forget information in the short-term memory. [51]

Retroactive interference then is the inhibition of retrieval of long-term memories by newer memories. In more intuitive terms, that means that learning new information directly leads to forgetting older memories. The effect takes place when any previously learned information is not rehearsed for a more extended period of time. Retroactive interference is the more frequent and more problematic of the two effects because it leads to the unlearning of information [64]. [28]

For the remainder of this paper, I chose the term "*information mutation*" to collect all these processes under a single, more intuitive word. It stems from the Latin word *mutatio* for change, or alteration and works well for the purpose of describing both decay and interference.

2.4 Multi-Agent Systems

As previously stated, the resulting prototype game from this thesis should simulate an interactive environment with NPCs that exchange and process information about the world state. Multi-agent systems (MAS) provide a suitable concept to interpret NPCs as autonomous, interacting agents.
A MAS is a computational model for simulating actions and interactions of intelligent, autonomous agents [100]. In the model, agents can represent both individuals and groups. The idea is that the resulting simulations can give a greater understanding of behavior systems or solve specific computational objectives. MASs are part of a growing research area and are applied in many different scientific fields such as ecology, economy, social sciences, biology, engineering sciences, physics, or, in fact, video games. Any field that is concerned with systems of many autonomous interacting entities can profit from MASs. Because of the ever more rapid advancements in computing power, these models can grow ever more complex, detailed and precise in structure and size, where previously, simpler models relying on less processing heavy assumptions were needed. [43]
The term multi-agent system is commonly used in areas such as engineering and artificial intelligence, whereas other sciences also refer to agent-based modeling [68].
In MASs, agents live in an environment. It is the surrounding space (either in the Cartesian sense or conceptual) in that the agents exist and perform actions. Environments can have different characteristics that are worth mentioning. An *accessible* versus an *inaccessible* environment allows the agent to obtain complete, accurate, and up-to-date information about the environment state. *Deterministic* and *non-deterministic* environments are different in the fact that in the former, any action has a single guaranteed effect, and there is no uncertainty about the resulting state of the environment. *Static* environments can be assumed to remain unchanged except by the acting agents' actions as opposed to *dynamic* environments. Finally, *discrete* versus *continuous* environments differ in the fixed and finite number of actions and percepts in them. [100]
The basic building blocks of MASs, the agents, can become complex models in themselves. They are worth a closer examination to understand better how NPCs can be modeled into agents.

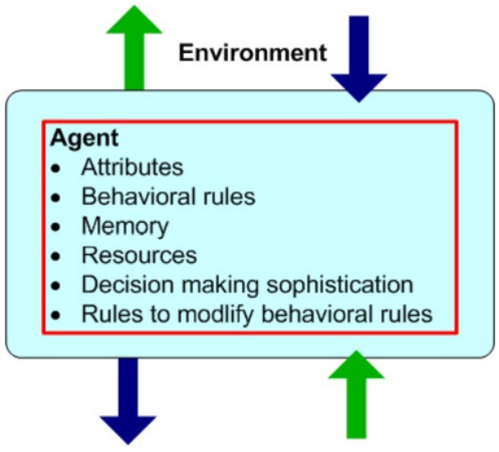

Environment
Agent
Attributes
Behavioral rules
Memory
Resources
Decision making sophistication
Rules to modlify behavioral rules

Figure 2.3: A typical agent [62]

2.4.1 Agents

Depending on the context the MAS is applied to, there exist various definitions for what precisely an agent is. For this thesis, I will adhere less to a concrete definition but to a set of properties that an agent must have to be considered as one:

- **Autonomy**: A agent must be able to function independently in its environment. Its interactions are not necessarily dependent on other agents, and it uses its capabilities to sense its environment to form decisions and actions.
- **Self-containment**: The agent has a definitive set of operations, attributes, and characteristics that make it a discrete individual entity. It has a systemic "boundary" that makes it possible to determine what is or what is not part of an agent and what might be a shared characteristic between agents.
- **Interaction**: To leverage the capabilities of a MAS, agents need to interact with each other. They have protocols that commonly include recognition of other agents, collision avoidance, information exchange, influencing other agents. These interactions can be particular to the MAS-specific domain.

Other properties for agents that could be considered as defining attributes can include *environmentally dependent*, which would mean that the agent's state is dependent on its situation in the environment, *having goals* that can describe objectives or measurements for the agent's effectiveness, *learning and adaption capabilities* to change behavior based on experience or *resource attributes* which track any metric of a resource an agent could earn, hold and spend. [62]
An example agent with its properties and indications of interacting with the environment and other agents can be seen in Figure 2.3.

2.4.2 Intelligent Agents

Agents with defined behaviors have a number of actions available to them that modify the environment around them. Not all actions can be necessarily performed at the same time. Usually, certain preconditions have to be satisfied, which might be highly depending on the current context or restrictions of the agent. Agents are equipped with *sensors* that are able to perceive the environment and changes in the same. An internal

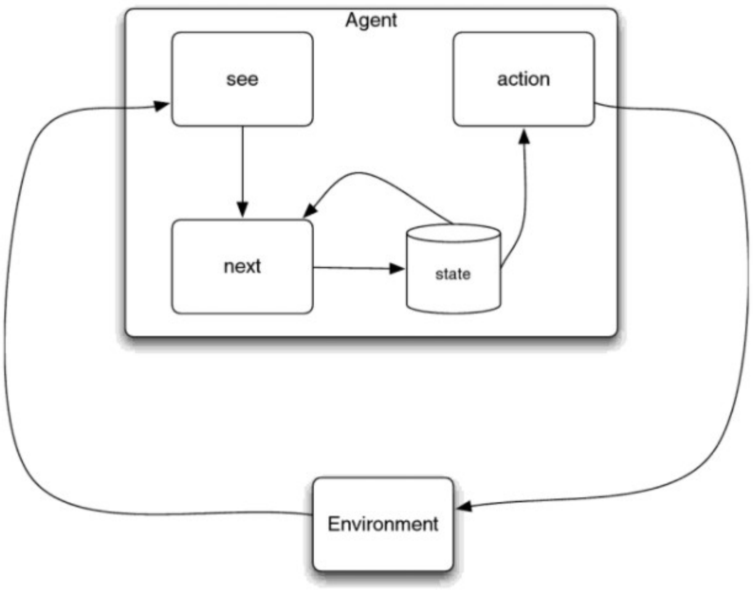


Figure 2.4: An agent as a state machine [100]

decision-making logic will determine the action to perform, and the *actuators* of the agent will then perform the action. This might then result in changes in the environment that other agents are again able to "sense" (Figure 2.4). The main issue when designing agents is to create their ability to decide *which* actions they should perform in a given context to satisfy their design objective best. [100]
This decision-making logic usually comes with the implication of "*intelligence*". For this thesis, I will follow Wooldridge and Jenning's list of "intelligent" characteristics of intelligent agents:

- **Reactivity**: the ability to perceive the environment and respond to changes that occur to satisfy design objectives.
- **pro-activeness**: select actions to be taken in a goal-directed manner. This is different from reactivity in so far as to change the agent's behavior before the fact.
- **Social ability**: the capability to use interactions with other agents to satisfy design objectives.

The characteristic that makes an agent "intelligent" are pretty demanding in their conception and implementation since they require logic that takes a changing environment into account. In scenarios with many agents that act in an environment at the same time, this becomes very complicated fast. [101]

2.4.3 State Machines

A practical approach to help with the design of intelligent agents is viewing them as state machines or agents with a state. State machines work on the basis of nodes that represent a distinct state a system is in coupled with state transitions that occur if specific conditions are met [46]. Each subsystem inside the agent can be seen as a state that the agent can take on. This helps a lot with the conceptual thinking of data transfer and the internal operations of the agent. It also allows for the employment of classical software

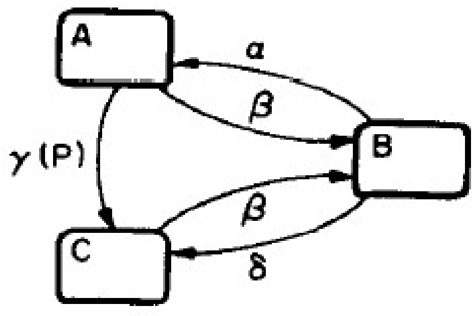


Figure 2.5: A simple state diagram [40]

engineering paradigms that are well tried and tested, and widely understood. [100]
There exist multiple standards for defining and visualizing the structure of state machines. They tackle the challenges of designing large and complex reactive systems. Since being *reactive* is one of the main characteristics of an intelligent agent, it makes sense to employ strategies that help with the design of such a system. [41]
A reactive system is characterized by being highly influenced by events outside and inside of the system. It is no trivial task to describe this kind of behavior in a way that is clear and realistic while retaining a sense of formality and rigorousness. Harel describes a visual formalism for designing and visualizing state machines that is based on box-shaped state nodes with arrows as state transitions (Figure 2.5). From this basis, Harel's work goes on to describe more complex visualizations for more extensive and more varied state machines, including behaviors like concurrency, clustering, or conditional transitions. [40]

2.4.4 Games as Multi-Agent Systems

It is quickly apparent how multi-agent systems could be leveraged to create a game world as an environment filled with autonomous NPC characters as intelligent agents.
The way game engines are built, they usually decouple subsystems like rendering, sound, physics, or user input from each other. Those systems are not easy to standardize, though, because they are oftentimes hardware-dependent and might not even need optimization for most use cases. The game logic is the subsystem in which game programmers implement the rules on which game mechanics are based. It is here where the heart of a video game beats. It is visible now that we can see a video game as a system and specific objects that act in the game as agents. Now, these agents need to adhere to the game's rules that are defined by the game designers. The game logic is thus implicitly related to the game objects internal processes that are associated with their autonomous agent's actions. The available actions are again dependent on the game mechanic, which forms the preconditions for state transitions that make actions available to agents. There now exists a link between game mechanics designed by designers to state transitions that define the decision-making of intelligent agents. The autonomous behaviors are also dependent on the communications between the objects of the game, either sending information about the game state or about the game object's state. [63]
This approach of realizing NPCs as intelligent agents in video games has not been explored often by developers yet. It is very likely that this would yield more compelling

gameplay, as I have described in earlier sections. We only slowly see more games in which agents actually interact with each other autonomously. For games in which interaction is simple, this would not be problematic to add. More complex interactions, however, can yield to unexpected and unwanted behavior of agents, especially if multi-agent interactions were not designed as an essential part of the game. Interactions between agents with conflicting goals could lead to two characters fighting whose interests would actually be aligned in the game world when they could actually find common ground on their goals and resolve the issue without conflict. There exists work on sharing, exchanging, and rejecting goals in traditional artificial intelligence studies [54]. The game developer community has yet to pick up this trend. This thesis aims to be a bridge between both fields. [24]

Computing power again becomes an issue when talking about decision-making for in-intelligent agents in games. Complex calculations that take up a lot of processing power might be needed to generate agent behavior. Video games already have high demands on computing power. The display of high fidelity graphics, the simulation of realistic physics, 3D audio playback, and network communications are only some of the expensive operations for both the processor and the graphics card. Developers, therefore, often have to reduce the complexity of their game logic algorithms which includes the internal logic of agents. Especially with multiple agents in a game that all form decisions in parallel, this can become a problem very quickly. In later sections, I will go into more detail about what my approach was to find a possible countermeasure for this problem for the prototype game of this thesis. Persistence is yet another issue that multi-agent system-based games would need to address. Serializing a current game state at any moment is not a trivial task as is. Add multiple interacting agents that all have their own state to interrupt and resume when the game is continued, and the result is a very complex problem. [24]

I have now described how NPCs can be defined as intelligent agents in a video game and the challenges this approach brings. For the more concrete scenario of the game mechanic that was implemented as part of this thesis, information exchange is at the basis of the prototype game. NPCs that share information about the game world and other agents need some form of a protocol of exchange. In the next section, I will describe how consensus protocols can be used to establish an understanding between agents.

2.5 Consensus

In complex computing systems, a tangible and fundamental problem is the problem of consensus. That means an overall ground truth that all participating agents reliably agree on even in the presence of faulty actors. There are numerous techniques and algorithms for coordination processes that result in agreeing on data values. Common real-world examples for consensus-dependent applications are committing database transactions, cloud computing, load balancing, blockchain, or clock synchronization. The fundamental consensus problem is the agreement of multiple agents on a single data value. During the operation of a system, some agents can fail or become unreliable due to errors. This means that consensus protocols need to be fault-tolerant. [94]

A basic approach to consensus is agreement on a majority value. What a majority is can again be different depending on the protocol. At least one more than half of the available votes are needed for a majority. Usually each agent gets one vote which also can lead to an incorrect consensus or none at all. A *quorum* is a popular choice for finding a majority.

2.5.1 Quorum

From the Latin noun *quōrum*, a quorum is the minimum number of votes an agent has to obtain in order to be allowed to perform an operation in the system. Quorum-based techniques ensure consistency in the system. The idea works as follows. Every agent in the system gets a vote v_i. The total number of votes in the system is V, and the accept and reject quorum numbers are V_a and V_r. The following rules must be satisfied in the implementation of the acceptance protocol:

1. $V_r + V_a > V$, where $0 < V_a, V_r \leq V$.
2. Before a process accepts, it must obtain an accept quorum V_a. That means the total of at least one agent that is prepared to accept and zero or more agents waiting $\geq V_a$.
3. Before a transaction rejects, it must obtain a reject quorum V_r. That means the total of zero or more agents that are prepared to reject or any agents waiting $\geq V_r$.

The first rule ensures that a vote cannot be accepted and rejected at the same time. The following two rules indicate the votes that an agent has to obtain before it can terminate one way or the other. Choices for quorum numbers are, of course, important decisions that influence the whole system. [74]

2.5.2 Consensus Protocols

Consensus protocols must follow some requirements in order to function correctly if they are expected to tolerate a limited number of failures in the participating agents' processes.

- **Termination**: eventually, every correct process decides some value
- **Integrity**: if all the correct processes proposed the same value x, then any correct process must decide x
- **Agreement**: every correct process must agree on the same value

System architecture varies greatly depending on the system domain. Consensus problems occur in all kinds of multi-agent systems independent of the attributes of the message exchange channels. Most communication models have authenticated channels meaning the participants are not anonymous. Senders and receivers of messages are known to the other respective participants. Some systems introduce a stronger, transferable form of authentication, where every participant "signs" the message so that the final receiver will know every processor from the original sender to itself. This form of authentication can tolerate a larger number of faults for the consensus protocol. [17]
In a consensus problem, there are only two types of failures for processes of participating agents:

- **Crash failure**: a process simply stops abruptly and does not recover
- **Byzantine failure**: a process appears to be failing, but it is unclear if it has actually failed. This includes malicious actions and attacks, which makes Byzantine failures the more disruptive class of failures. It is named after "*The Byzantine Generals Problem*" [55].

A consensus protocol tolerating Byzantine failures must be able to fend off every possible error that can occur. [17]
Synchronicity is another vital aspect of consensus protocols. To approximate the asyn-

chronous nature of real-world communication, systems are often modeled as synchronous. In synchronous systems, communication happens in rounds. A process sends all its messages in one round and then receives all messages from other processes. Therefore, no message is allowed to influence another message in the same round. [17]
Usually, there is a configuration process prior to any communication that permissions agents to take part in the communication. If no such authentication of agents takes place, the system is open to attacks that simply create enough virtual participants that vote in the attacker's favor, so-called "*Sybil Attacks*". There are, however, permissionless protocols that allow anyone to join the protocol. For those systems, there is another cost or barrier involved. Bitcoin is, of course, famous for introducing the first permissionless consensus protocol that uses Proof of Work to earn the right to participate [38]. There are other permissionless consensus protocols such as Proof of Stake, Proof of Space, Proof of Authority, or Proof of Personhood, which is a promising approach for consensus among NPCs. [17]

2.5.3 Proof of Personhood

Going forward, I will view the system of NPCs and how they exchange information as an asynchronous, permissionless consensus problem. Asynchronous because characters only exchange information when they encounter other characters or events in the world that create a piece of information and permissionless because, in a "social" group of NPCs and a player, there is no specified process for authentication before taking part in the information exchange. Proof of Personhood is a technique that has humans at its center to ensure authenticity. [8]
The idea is to guarantee each unique participant an equal amount of votes independent of economic investment. This differs significantly from other permissionless consensus proto- cols like Proof of Work or Proof of Stake, where voting power is given proportionally to an investment from the participant. Social and democratic tendencies are clearly apparent to have a part in this approach. Blockchain systems that are more or less established now, like Bitcoin and Etherum, cannot possibly claim social fairness since a comparably small number of big players can hold and thus control choices on the blockchain. [86] The previously mentioned Sybil attacks are a central problem for identity-based protocols. They create a large amount of fake virtual identities to gain a controlling majority in a system [26].
This "unique human" or "semi-unique human" problem has resulted in much research towards network protocols that make use of subjective inputs. Voting, vouching, or interpreting are measures taken to create a Sybil-resistant consensus for identity to create fairer, more social digital networks for humans. [91]
In order to guarantee reliable protection from Sybil attacks, every single identity in the network needs to be *unique* and *singular* to ensure that no two people have the same identifier, and no person should be able to have more than one identifier.
There are two additional requirements for Proof of Personhood approaches: The first is *self-sovereignty* which ensures that anybody can create and control an identity without any influence of a centralized third party like authorities or institutions. The second is being *privacy-preserving* which means that every participant can acquire and use their identifier without revealing personal information that would allow the inference of the identity in the process.
Proof of Personhood protocols tackle these three requirements by utilizing the following concepts:

- **Subjective substrate**: Proof of Personhood needs some form of "task" to replace the computational work of Proof of Work or the financial stake of Proof of Stake protocols. Such a substrate would need to be easy for humans to produce but difficult for artificial intelligence. Furthermore, it would need to be challenging to produce for humans *twice* while involving zero or absolutely minimal personal information.
- **Objective incentive**: There needs to be an incentive to participate strong enough so that it is more valuable to be a part of the network as a legitimate agent than as a fake identity.

Some examples of Proof of Personhood applications include CAPTCHAs to prove that a user is a human or a very basic approach of so-called "pseudonym parties". They build on the fact that humans can only physically be at one place at a time. [91] In this method, humans physically meet at a specific location and time to confirm their authenticity at the meeting, for example, by scanning QR codes that create anonymous credentials [8].
Part of the prototype game was finding a suitable subjective substrate for NPCs to base their information exchange on. I designed a heuristic for information believability that NPCs take into account when receiving a new piece of information.

2.6 A Heuristic for Information

The term "heuristic" comes from the Greek *εύρίσκω*, which translates to "I find" or "I discover". Heuristics are techniques that approach exact results rather than precisely calculating them. Any method that reaches a sufficient approximation for a given problem in less time than traditional or correct methods can be described as a heuristic. [47]
In computer science, heuristic functions also approximate an exact solution for otherwise potentially expensive calculations [76]. For the prototype game of this thesis, I came up with a heuristic that allows NPCs to evaluate incoming information with regard to their believability. It is a calculation that should simulate the cognitive process of "thinking about" something that a character learns. Even though it is an approximation of such a process, it is quite an expensive calculation.

2.6.1 MapReduce

MapReduce is a programming model that allows the distribution of calculations on big data sets to reduce the workload for single processes. I applied the *MapReduce* concept while designing the heuristic function so that results become available in a reasonable time.
The distribution of large datasets using a large number of computers is collectively referred to as a cluster or a grid. While this process often appears inefficient compared to algorithms that are more sequential, *MapReduce* can be applied to significantly larger datasets than a single server can handle. For example, a large server farm can use *MapReduce* to sort a petabyte of data in only a few hours [19]. *MapReduce* frameworks consist traditionally of the following three operations:

1. **Map**: each processing node applies a *map function* to a local dataset, the result is written to temporary storage. In a MapReduce framework, a master node ensures only one copy is processed, and no redundancy occurs. Map functions, as the name implies, "map" the input data to an output key.

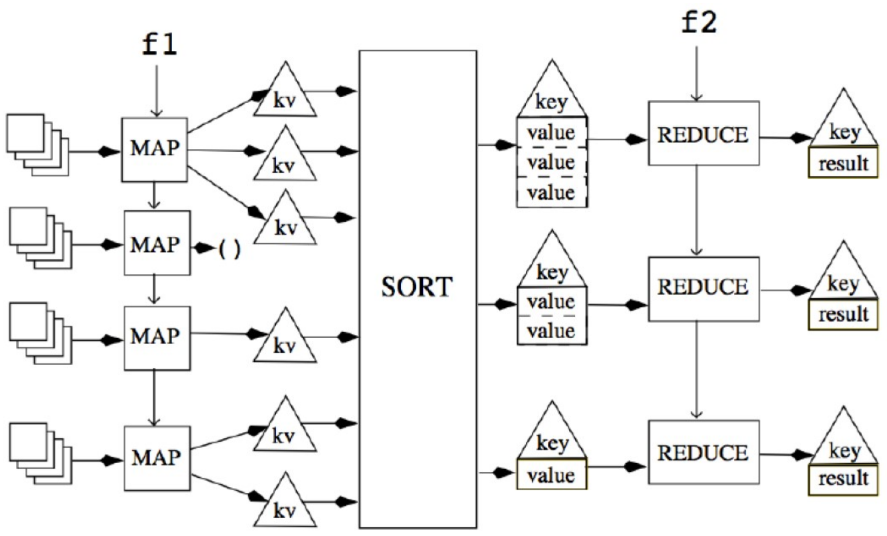


Figure 2.6: A MapReduce framework with a map function $f1$ and reduce function $f2$ [22]

2. **Shuffle/Sort**: Data is redistributed based on output keys. Results with the same output key are transferred to the same nodes for further processing.
3. **Reduce**: The collected groups of data are now processed in parallel on the respective nodes. This is performed using aggregate operations that "reduce" the group to a single value.

MapReduce allows distributed processing of the map and reduction functions. The mapping can be performed in parallel if the operations are independent of each other. Similarly, the reduction phase can be processed in parallel as well. For this to function properly, all outputs of the map operation that share the same key need to be presented to the same reducer at the same time, or the reduction function needs to be associative. Figure 2.6 shows a schematic view of a MapReduce framework. [22]

2.7 Inference Engines

I have established that NPCs will "think about" and evaluate the information they receive. Additionally, they also have a routine that allows them to "reflect" on information they already possess. According to a specified set of rules, agents can "deduce" new information in the form of: "If I have information A and information B and the rule $A \wedge B \Rightarrow C$ exists, then I will learn information C." Logic inferences like this can be produced with so-called *inference engines*. [42]
For an inference engine to work, a knowledge base needs to be present. The collective information set of an agent can represent such a knowledge base. The inference engine applies logical rules to the knowledge base and deduces new knowledge. This process is executed repeatedly until no new knowledge is deduced since each new fact could trigger new rules to apply.
There are two modes, rules that can be applied in an inference engine: *forward chaining* and *backward chaining*. Forward chaining takes the known facts in the knowledge base and creates new facts based on the inference rules. Backward chaining starts by looking at the goals and then determines the facts that need to be asserted for the goals to be true.

An example would be the following inference rule in pseudo-code:

$$\mathit{isAgentArmed}(a) \land \mathit{isAgentEnemey}(a) \Rightarrow \mathit{isAgentDangerous}(a) \tag{2.3}$$

The rule states that if an agent a is both *armed* and an *enemy*, then the agent should also be considered *dangerous*. Forward chaining would look at each fact in the knowledge base and search for facts for an agent a that state that said agent is *armed* and an *enemy*. The rule would then add the fact that agent a is also *dangerous*.
Backward-chaining would look at the result *dangerous* and then look for the prerequisites *armed* and *enemy* to then determine if agent a is in fact *dangerous*. Although implementation might sound similar, the conceptual approach to the two modes is different. [31]

2.8 Related Work

Even though the proposed game mechanic is novel in that it leverages multi-agent systems, there has been some work in similar fields, some of which I want to mention at this point.

2.8.1 Emotion Engines

In his Master thesis, Mizra explores emotion engines and how they can be applied to strengthen the narrative of video games. Emotion engines, in general, are computational models for approximating human emotions. Mizra provides a standardized way of adding emotional components based on any emotion theory and software engineering principles. This helps in designing an architectural paradigm for an emotion-oriented approach to game development. The framework also provides a way to generate an adaptive narrative not only for the story of the game but also other visual elements of the game. He defines the emotional states of agents as statecharts and personality as a mapping function. In the framework, events in the game world come with an emotion package that alters the emotional state of involved agents. This is a lightweight and very platform-agnostic approach that would be very valuable to add to the prototype of this thesis. [65]

2.8.2 Multi-Agent Systems and Sandbox Games

Ocio and Brugos, in their paper on multi-agent systems and sandbox games, propose to start new research aimed at achieving more realistic sandbox games, building cities full of life, using, improving, or creating specific algorithms or techniques built upon existing work relating to multi-agent systems. They argue that sandbox games are suitable to be viewed as environments for intelligent agents. They furthermore provide an ar- architecture for agent-environment interactions as well as agent-agent interactions and an interface on how messages between entities could be structured. There has been much research done into developing intelligent agents, but not much of that research has been taken over into video game development. [73]

3 Implementation

The following chapter describes the prototype game I created as part of this thesis. I explain how I applied the concepts and research introduced in the previous chapter.
The game simulates a small medieval kingdom and its inhabitants. The player controls a knight who is sent on a quest by the queen. The king is missing, and the player shall return the king to the castle. The player can move around and interact with non-player characters (NPCs) and some key items. First, I will introduce the *Information Model* that I created to define information in the game. Then I will go into detail about the *agents* that inhabit and exist in the game world. After that, I will describe the *game world* itself, followed by a *gameplay* description. Then I will focus on the *dialogue feature* and how information can be retrieved and introduced into the system. Following that, I will give an explanation of how narrative is introduced using *quests*. Additionally, I will provide the concrete implementations of both the information *heuristic* and the *consensus protocol*. The chapter will be closed off by discussing the *narrative implications* of the described gameplay.

3.1 Technology

Before I go into detail about the game logic, I want to explain my choice of the development environment. For the prototype game, I chose the Unity engine developed by Unity Technologies [96]. This choice was made because I have multiple years of experience with the Unity engine, and it allows for fast prototyping that is easy to iterate. Unity is widely used across the video game industry, has an active developer community, and is well documented. The programming language for Unity is Microsoft C# with the .NET framework, which I also have many years of experience with. All assets that are used in the prototype game are either purchased or free to use. All asset creators are credited adequately in the prototype. The project is available under the MIT license at `https://github.com/doingitraith/Story-generation-with-mutli-agent-systems` [48].

3.2 Information Model

At the center of the game mechanic created for this thesis is the *Information Model*. I propose a conceptional model for information objects that encapsulate statements about semantic and factual game world data (see 2.3). The statements encapsulated in an `Information` object are primarily characterized by its *Verb*. The verb is the primary indicator of the assertion of the information content. Every information object also contains a *Subject* that denotes who or what the information statement is concerned with. The third part of the information is dependent on the verb and can either be an *Object*, an *Adjective*, or a *Location*. Thus every information object *I* consists of a triple of information elements:

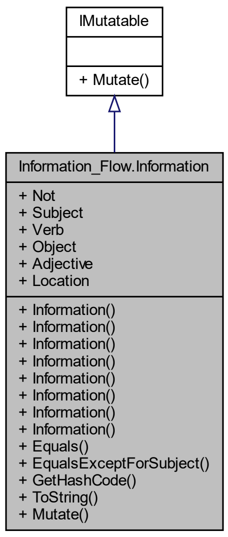


Figure 3.1: Inheritance diagram of the Information class

$$I := \{s, v, x\}$$

Where s is the subject, v is the verb, and $x \in \{\textit{Object, Adjective, Location}\}$ depending on the verb. Figure 3.1 shows the class diagram for the `Information` class. Although the elements of an `Information` object refer to actual game objects in the game world, this link is purposely severed when an `Information` object is created. This is done so that the information statement is actually just that and has no exploitable connection to other objects anymore. Thus the *Information Model* and the programming interface remain decoupled, which allows for a more accurate simulation.
The `Information` class also implements the `IMutatable` interface, which only declares the function `Mutate`. This function should reduce the accuracy in the statement by reducing the accuracy in any of the statement components every time it is called.

3.2.1 Verbs

Verbs in the *Information Model* come in three types:

- **IS**: Creates information statements about the *state* of a subject of the form "*Subject* `IS` *Adjective*".
- **HAS**: Creates information statements about a possessive relationship between subject and object of the form "*Subject* `HAS` *Object*".
- **AT**: Creates information statements indicating the location of a subject of the form "*Subject* `AT` *Location*". The `AT` type presents a special form of the `IS` type, but in the context of a traversable 3D world, locations represent such an important piece of state that its own information type is justified.

Verbs are instances of the `Verb` enumeration. The constructors in the `Information` class are designed in such a way that the verb is set by the constructor, and there exist multiple overloads to create the different types of information objects (Listing 3.1). This ensures that the developer cannot easily create invalid `Information` instances.

```
public Information(Agent agent, Item object)
    => (Subject, Verb, Object, Adjective, Location, Not) =
    (agent.InformationSubject, InformationVerb.Has, object.
        InformationSubject, null, null, false);

public Information(WorldObject subject, InformationAdjective
    informationAdjective)
    => (Subject, Verb, Object, Adjective, Location, Not) =
    (subject.InformationSubject, InformationVerb.Is, null,
        informationAdjective, null, false);

public Information(WorldObject subject, InformationLocation
    informationLocation)
    => (Subject, Verb, Object, Adjective, Location, Not) =
    (subject.InformationSubject, InformationVerb.At, null, null,
        informationLocation, false);
```

Listing 3.1. Information class constructors

This way, `Information` objects can be created with only the content in mind, and the constructor sets up the object correctly for usage.

3.2.2 Subjects

As mentioned, every piece of information contains a *subject*. They represent the concerning entity of an information statement. A *subject* is an instance of the `InformationSubject` class that conceptually either represents an `Agent` or an `Item`, both of which inherit from the `WorldObject` class.
What exact type of object a *subject* is in a given `Information` object is again dependent on the *verb*. For example, there can be no information of the form "*Item* `HAS` *Agent*" because an item cannot own an agent. The constructors for `Information` objects make sure that no erroneous object can be created.
Subjects consist of a name that denotes the *subject* and two boolean values that state whether the *subject* is an agent or not and whether it is a unique world object (Figure 3.2).
Subjects also have a `Mutation` object attached that is responsible for decaying the accuracy of the *subject*. I give a detailed explanation of mutations in Section 3.2.5.

3.2.3 Adjectives

An *adjective* represents a detail or characteristic of a *subject's* state. They can be arbitrarily defined by a developer. They represent the properties of a *subject*. *Adjectives* are instances of the `InformationAdjective` class (Figure 3.3). In the *Information Model*, there are two differentiable types of `InformationAdjective` objects:

- `InformationProperty`: Represents factual information about a *subject*. For example, "*Agent* `IS` *alive*." is a factual piece of information that can either be true or false.

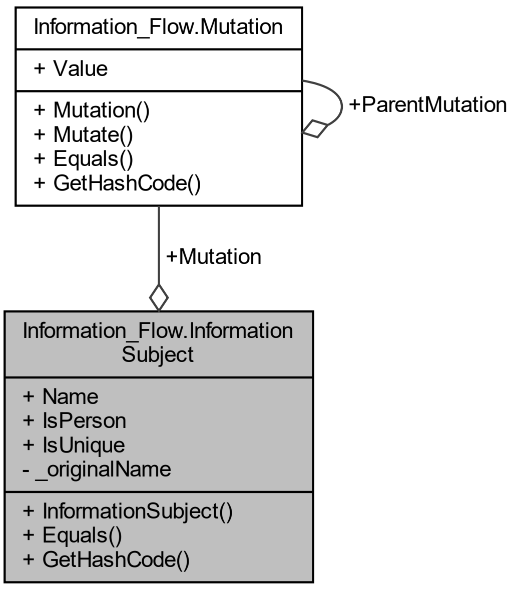


Figure 3.2: Inheritance diagram of the InformationSubject class

- `InformationOpinion`: Represents statements about a *subject's* state that are influenced by an information holder's relationship to the *subject*. For example, the statement "*Agent* `IS` *dangerous*." can be true for one agent but false for another.

This distinction is introduced so that conceptually there can be a difference between facts and opinions that someone has about any *subject*.
An `InformationAdjective` consists of the textual *Characteristic* and a list of *contradictions*. The contradictions are other *adjectives* that conflict with the *adjective* they are attached to. A simple example would be the two information statements "*Agent* `IS` *alive*." and "*Agent* `IS` *dead*.". Both cannot be true at the same time. Therefore, each *adjective's* contradictions list contains the respective other. When an agent learns a piece of new information that contains an *adjective*, they check the *adjective's* contradictions and resolve any conflicts.
At the start of the game, an initial list of *adjectives* that exist in the world is created. After that, the contradictions lists are created and attached to each *adjective*.

3.2.4 Locations

As mentioned before, a *location* is a special kind of *object* for the `AT` information type. In a 3D game world that is driven by a multi-agent system, agents need to move around. For that, the `InformationLocation` class attaches a position in the world to the information so that agents can retrieve an actual location from the information statement.
A *location* also has a `Mutation` object attached to it, so its accuracy can be reduced. Other than that, the `InformationLocation` has only a textual name and the 3D position as a public interface.

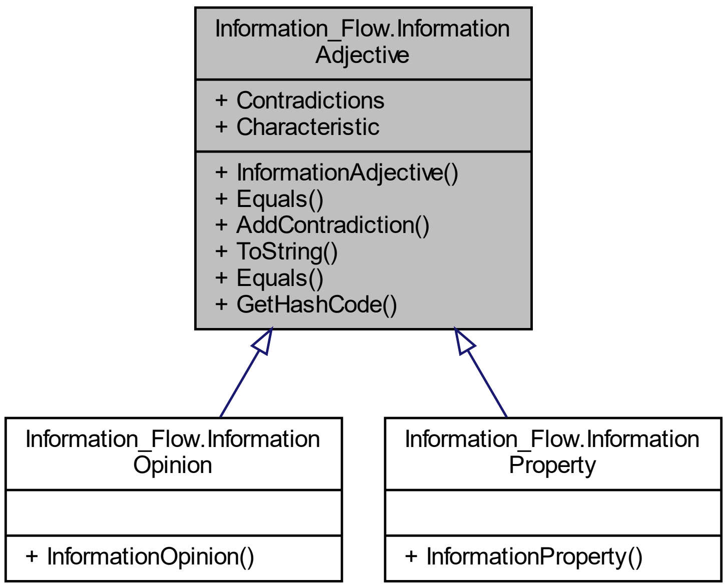


Figure 3.3: Inheritance diagram of the InformationAdjective class

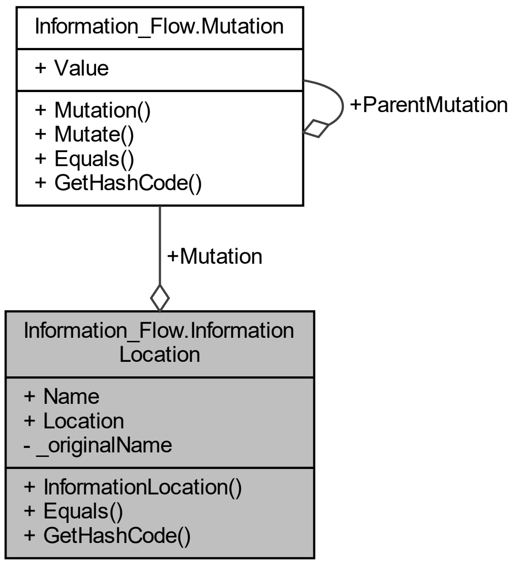


Figure 3.4: Inheritance diagram of the InformationLocation class

3.2.5 Mutation

To simulate the decay of information and agents forgetting details, the *Information Model* implements a *Mutation* functionality. `InformationSubject` and `InformationLocation` objects have an object of the `Mutation` class. These represent a hierarchy of less precise information values. Every `Mutation` object has a reference `ParentMutation` that points to a `Mutation` object that holds the next less precise information value.
As an example, we assume the `InformationLocation` l with the name value "*Village Square*". The `Mutation` object can point to a less precise information value "*Village*", which in turn points to an information value "*Somewhere in the South*". This way, when l is mutated once, its name value will be "*Village*" and, if mutated again, will decay to "*Somewhere in the South*" where it will remain.
This mechanic allows for agents to slowly forget about details in their information statements which makes the collection of memories more chaotic over time.
The values for the mutation hierarchy are set by the developer before the game starts.
The `Mutate` function in the `Information` class that is provided by the `IMutatable` interface triggers the mutation of any of its statements' components.

3.3 Agents

In any multi-agent system, the central component is, of course, the agents. In the prototype game, those agents represent the NPCs who live in the game world or the player (Figure 3.5). The only difference is that the player agent is controlled by the player instead of the game. However, for the NPCs, there is no difference between the player agent or any other NPC. In the following, I will generally talk about NPCs as agents for a better understanding of the conceptual implications in a multi-agent system. Agents move around in the world to perform actions, exchange information, and change their state depending on what they learn. The `WorldObject` class is the parent class for both `Agent` and `Item` classes. It represents physical entities that exist as part of the multi-agent system. `WorldObject` itself is a child class of Unity's `MonoBehaviour` class which is the base class of all game objects managed by the Unity engine.
All `Agent` instances inherit the members of their parent classes and thus have a textual identifier *Name*, a *Location* to denote the object's current location, exposed properties to set up the *InformationSubject*, and an integer value *WorldImportance*. This value is essential for the information heuristic for believability which is described in Section 3.8.
WorldImportance indicates the initial interest a piece of information gets through the involved object. This interest can be positive or negative. A piece of information about the castle might have high positive value since the castle is an important place in the kingdom. Information about an enemy bandit, however, will have a negative number, as people are afraid of the bandits. These values should be an objective measure of an object's interest value. Subjective views from the agents are taken into account when the heuristic is calculated.
The critical properties of the `Agent` class are the *Inventory* which stores items in the agent's possession, *Acquaintances* containing other agents that have interacted with it, *ImportantPeople* are agents that have a close relationship with the object like friends or loved ones and *Quests* which is a list of objectives the agent has. Through the *ImportantPeople* property, specific agents can be treated differently and more favorably than others. This allows the introduction of a social network component into the multi-agent system.

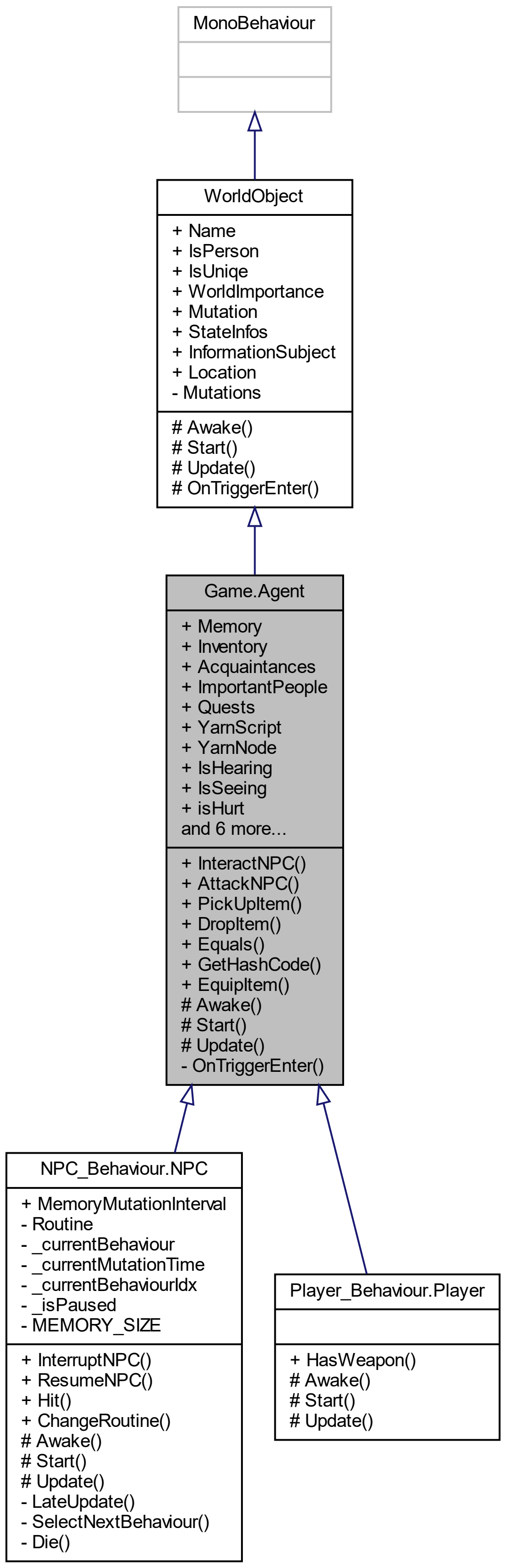


Figure 3.5: Inheritance diagram of the Agent class

The boolean values *IsSeeing* and *IsHearing* allow the determination what kind of information the agent is able to "sense" respectively.
The heart of the game mechanic is the *Memory* property. It is an `InformationManager` object. As the name implies, it stores, processes, and manages all the incoming pieces of information.
The different actions an agent can execute are also defined here. This includes interacting with other agents, picking up and dropping items, or attacking agents.

3.3.1 NPC

The `NPC` class is a child class of the `Agent` class and defines additional properties and functions that are unique only for NPCs.
This includes a *Routine* property which is a list of behaviors the agent will execute one after another to simulate a daily life of a human. When they reach the end of their routine, they will start again from the beginning.

3.3.2 Player

In the scope of this thesis, the `Player` class behaves very similarly to other agents. It has components attached that enable input to control the player agent. It needs its own class so that developers can differentiate between NPCs and the player. This is necessary as the agents might need to behave differently when interacting with the player agent to expose more information to the actual player in front of the screen. For example, when a dialogue is started, an NPC will not simply send information and continue with its routine, but it will pause, and the dialogue interface will be opened where the information exchanged is controlled by the player.

3.3.3 Behavior

In order for the intelligent agents to move, interact and follow their objectives, I defined multiple behaviors that allow them to perform actions (Figure 3.6). The following behavior classes exist:

- `ExchangeInformationBehavior`: This behavior allows two agents to engage in an exchange of a piece of information. They choose which information to send each other based on the information heuristic.
- `SendInformationBehavior`: As a child class of `ExchangeInformationBehavior`, this behavior selects a random agent from the list of *ImprotantPeople* and sends them a piece of information also based on the information heuristic. This behavior can be interpreted as "sending a message" to a distant friend. I implemented this so that information would also reach farther parts of the world eventually.
- `WalkBehavior`: This simply allows the agent to move to a specified location in the world.
- `TalkBehavor`: This would allow NPCs to stop the player and engage in a conversation with them. In the prototype game, this behavior is never used, and it is only the player who can initiate a conversation.

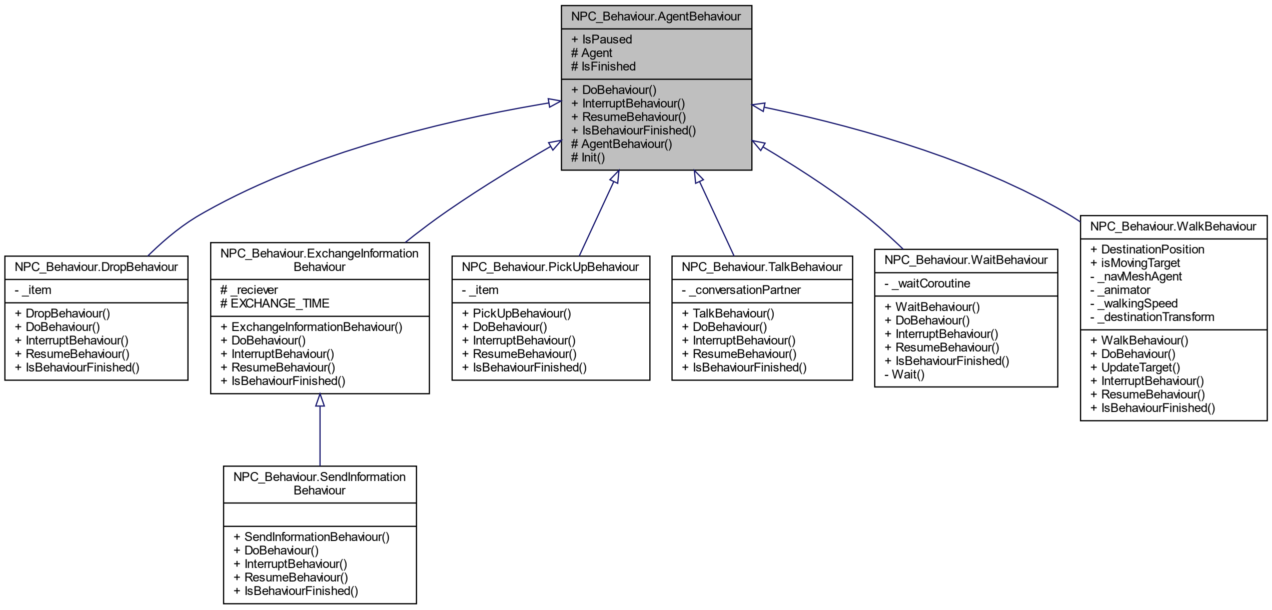


Figure 3.6: Inheritance diagram of the AgentBehavior class

- `WaitBehavior`: To simulate NPCs that are waiting or "doing nothing" for a specified time, I implemented this behavior. It is mainly to make the simulated behavior look more realistic and have NPCs stand in a place for a while.
- `PickUpBehavior` and `DropBehavior`: These two are implemented to let NPCs pick up a specified item in the vicinity or drop it in front of them, respectively.

The parent `AgentBehavior` class is an abstract class and declares functions for starting, interrupting, and resuming a behavior. Additionally, it declares the function `IsBehaviorFinished` to check if the behavior is completed. This simple setup of behavior allows for easy extension of the capabilities of agents.

3.3.4 Information Manager

At the heart of the agents' logic is the `InformationManager` class. It provides all the necessary functions to add, manage and process incoming `Information` objects for an agent. Its main tasks are adding new pieces of information to the memory and evaluating them on a regular basis to update the information heuristic values and the trust values of an agent's known associates.
It manages the different pieces of information with lists of `InformationContext` objects. Those are objects that pack an `Information` object with contextual data about how the information and how it was received. The context object stores how many times the information was received already by distinct other agents, whom it was received from, the time since it was last received, and current values for believability and information heuristic. This additional contextual data is used by the *Information Manager* to calculate current heuristic, believability, and trust values.

The *Information Manager* provides two lists of `InformationContext` objects to manage its memories:

- *Speculative Memory*: the speculative memory stores every piece of information that an agent receives. It is a list of unconfirmed information that needs to be evaluated before it can be trusted. Only information that the agent did not perceive itself but was given by others is considered speculative.
- *Stable Memory*: confirmed or trusted information is stored in the stable memory. Everything an agent learns by its own senses, like seeing, hearing, or otherwise witnessing, goes directly to the stable memory. If information is received via another agent, it needs to be verified through believability before it is transferred from the speculative memory to the stable memory. When an agent evaluates the information stored in its memory, it also updates believability values for every piece of information. This value is closely tied to the information heuristic. If the believability of information is above a certain threshold, it is transferred to the stable memory, and if it falls under the threshold, it is sent back to the speculative memory.

This structure of speculative and stable memory is taken from high availability backup systems. A process called chain replication uses a similar technique, where any change to the system is put into a speculative history until all nodes in the backup change have confirmed it. After that, it is transferred through all nodes again but in a stable history. When all nodes have confirmed the change in the stable history, the update was successful. [80]
Although agents are only a single node, I tried to replicate this confirmation methodology with speculative and stable memories.

3.4 Game World

The agents in the prototype game live in an environment. The environment in the prototype game is *inaccessible*, *deterministic*, *static*, and *discrete*. It is inaccessible because agent learn of changes in the environment state only when they perceive or receive information about them, not as they happen. It is deterministic because every change an agent performs on the environment's state is known beforehand. The environment is static because the topology and structure do not change in this prototype, and it is discrete because the space agents can move in is limited.
In the prototype game, the environment for the multi-agent system is the game world. I created a 3D world that contains interesting and memorable locations that agents can move around in. I recreated a small kingdom with several settlements where different agents live and go about their day. The key locations are *the village*, *the castle*, *the sawmill*, *the forest*, *the lake*, *the farm*, and *the enemy camp* 3.7. The landscape is filled with paths and landmarks to make orientation easier and guide the player between the locations.
I chose a medieval kingdom for my environment since it is a well-understood and established context for video game worlds. Ford, in his work about medieval towns in single-player games, describes how the boundaries between village and nature are also boundaries between the home and the frontier, the old and the new, the familiar and the unknown. This does not only relate to classic narratives about heroes venturing out into the wild but also says interesting things about this dichotomy in game worlds. Video games often contextualize towns and nature through their mechanics. There are games, for example, where you cannot use your weapons while inside a town or city. They also

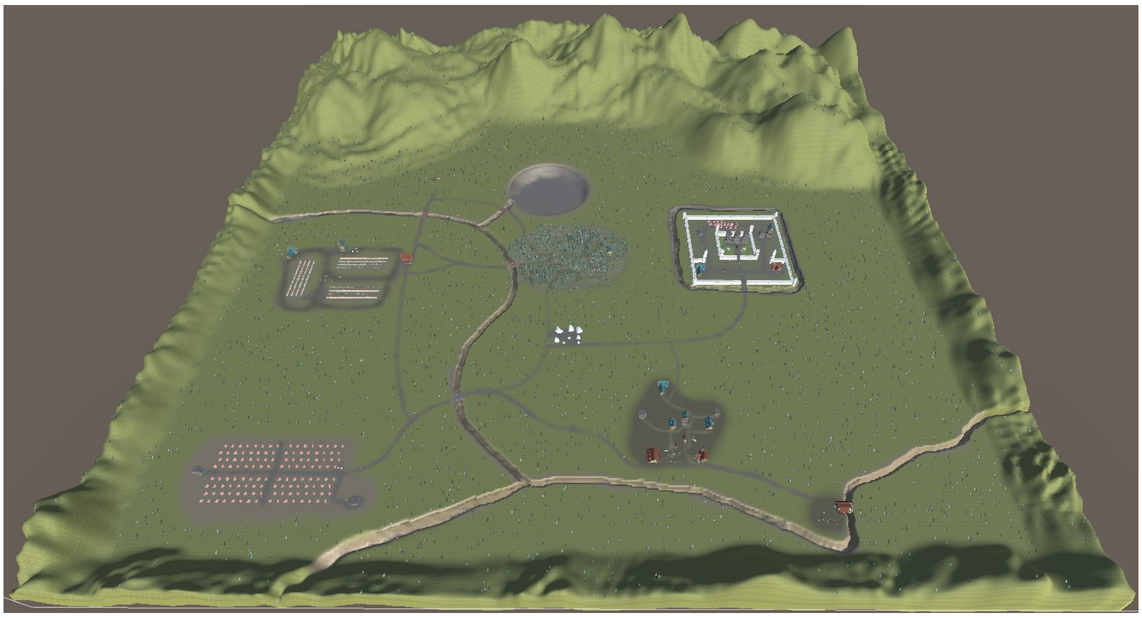


Figure 3.7: An overview of the game world.

very often serve as quest hubs or quest destinations. This means that the borders define on a mechanical level what the player is allowed to do. Much like with real-world definitions of borders, this might say more about the player and their position in the game world than it does about the city or town itself. [35]

3.5 Gameplay

I have touched previously upon what the player interactions are that form the gameplay of the prototype game. Players control a knight in the kingdom who sets out to go on quests and adventures in the kingdom (Figure 3.8). The player controls the movement of the knight and can talk to NPCs and interact with key items. If the player picks up a weapon, the character is able to attack other agents.
The novel game mechanic this prototype now presents, and that is the topic of this thesis, is more about the *How* than the *What*. Initially, the player character is given a quest to return the king back to the castle. That is all the information the player is given who is now free to roam the game world to try and reach this goal.
Through talking to NPCs and learning new information about the world, the player can find clues that help them reach their goal. Where exactly is the king? Is he guarded by a monster? How can I defeat the monster? NPCs can give pieces of information that let the player step by step deduce an answer to this question.
Alternatively the player can go a completely different route. By talking to NPCs, they can introduce new or even wrong pieces of information into the world. Could I convince enough people that I am the king? If someone would tell the queen, would she believe I am the king? Moreover, would that allow me to go back to the castle and thus achieve the goal?
The systems that I have created for this prototype allow for varied and interesting solutions to quests, even for the little amount of content that is in the prototype.

Figure 3.8: A sample gameplay scene.

3.6 Dialogue

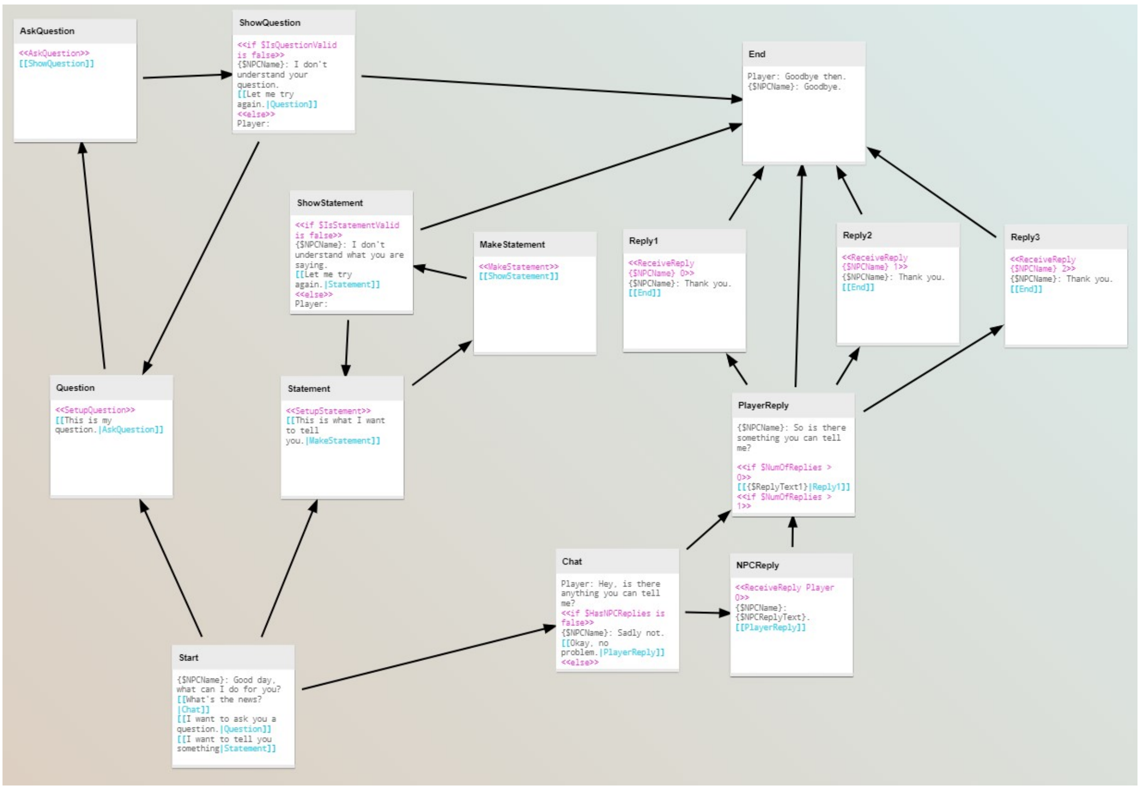
The primary way to purposely learn and give out information in the game is to talk to NPCs. For that, I have created a dialogue system that lets you either receive a piece of random information from an NPC, ask a specific question, or make a statement on your own.
I used Yarn Spinner from Secret Lab to create the dialogue [85]. It is a lightweight tool that comes with a Unity plug-in and allows for branching node-based dialogue, and supports variables and simple logic in its dialogue files.
Yarn Spinner comes with an editor that easily allows developers to set up and write complex and branching dialogue files.

3.6.1 Questions

Allowing the player to ask questions about specific game objects is much more goal-oriented than randomly hoping for helpful information to reach the player. The dialogue interface allows the player to create questions from a question verb, an information verb, and an object(Figure 3.10). The question verbs *Who*, *What*, *Where*, and *How* indicate whether the expected answer will be an NPC name, an item, a location, or an adjective. The question object contains options depending on the selection of the question verb and information verb. If the NPC has a piece of information that matches the question requirements, it will tell it to the player.

3.6.2 Statements

This allows the player to create new information based on objects they have encountered while playing. In the dialogue interface, the player can create information statements like they are defined in the *Information Model*(Figure 3.11). The selected verb will define what



```
AskQuestion
<<AskQuestion>>
[[ShowQuestion]]

ShowQuestion
<<if $IsQuestionValid
is false>>
{$NPCName}: I don't
understand your
question.
[[Let me try
again.|Question]]
<<else>>
Player:

End
Player: Goodbye then.
{$NPCName}: Goodbye.

ShowStatement
<<if $IsStatementValid
is false>>
{$NPCName}: I don't
understand what you are
saying.
[[Let me try
again.|Statement]]
<<else>>
Player:

MakeStatement
<<MakeStatement>>
[[ShowStatement]]

Reply1
<<ReceiveReply
{$NPCName} 0>>
{$NPCName}: Thank you.
[[End]]

Reply2
<<ReceiveReply
{$NPCName} 1>>
{$NPCName}: Thank you.
[[End]]

Reply3
<<ReceiveReply
{$NPCName} 2>>
{$NPCName}: Thank you.
[[End]]

Question
<<SetupQuestion>>
[[This is my
question.|AskQuestion]]

Statement
<<SetupStatement>>
[[This is what I want
to tell
you.|MakeStatement]]

PlayerReply
{$NPCName}: So is there
something you can tell
me?

<<if $NumOfReplies >
0>>
[[{$ReplyText1}|Reply1]]
<<if $NumOfReplies >
1>>

Chat
Player: Hey, is there
anything you can tell
me?
<<if $HasNPCReplies is
false>>
{$NPCName}: Sadly not.
[[Okay, no
problem.|PlayerReply]]
<<else>>

NPCReply
<<ReceiveReply Player
0>>
{$NPCName}:
{$NPCReplyText}.
[[PlayerReply]]

Start
{$NPCName}: Good day,
what can I do for you?
[[What's the news?
|Chat]]
[[I want to ask you a
question.|Question]]
[[I want to tell you
something|Statement]]
```

Figure 3.9: The NPC Yarn dialogue file

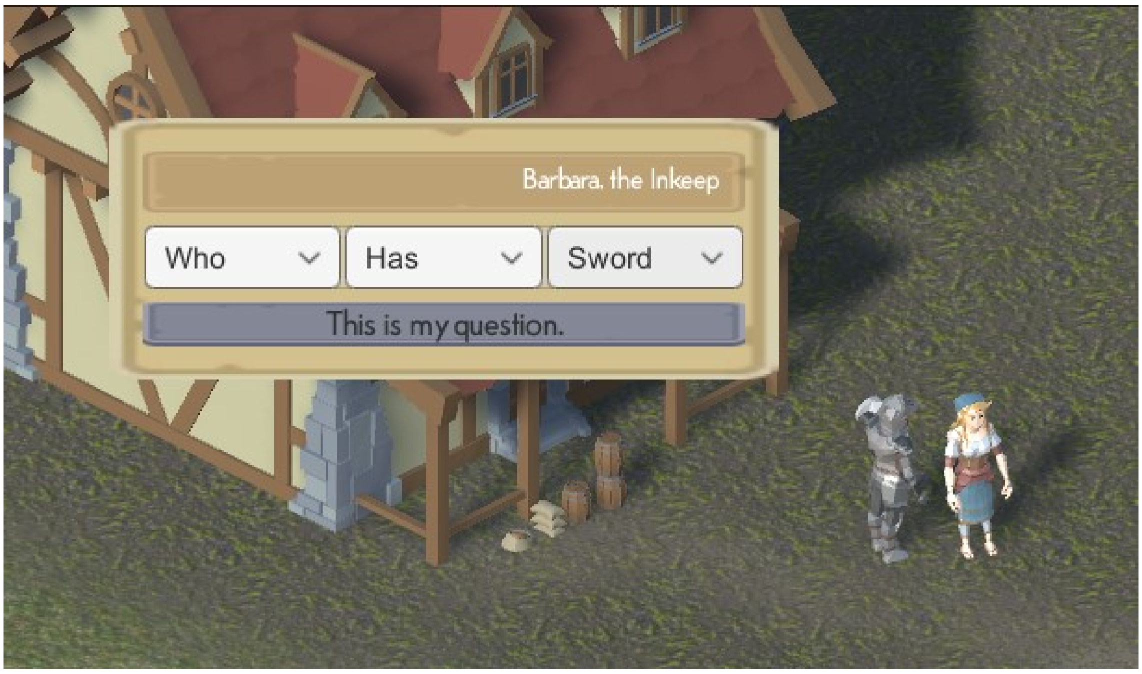


Figure 3.10: Posing a statement to an NPC.

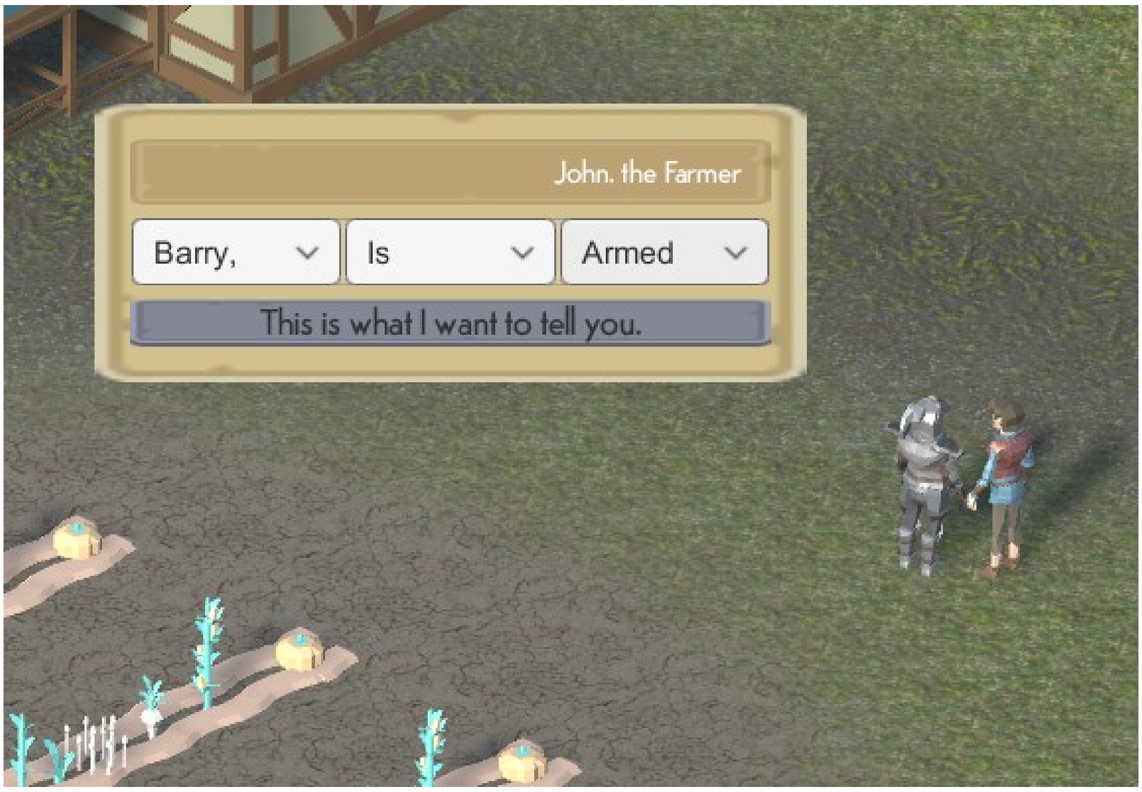


Figure 3.11: Asking a question to an NPC.

options are available for the information subject and information object. The available subjects and objects depend on the NPCs, items, and locations the player has encountered so far. Only adjectives are universally known by all agents as they do not represent things in the world but rather concepts.
If the information statement is well-formed according to the *Information Model*, the newly created information will be transmitted to the NPC as if any other agent would have told to it.
The functionality to create any information from known components creates the ability to lie for the player, which creates exciting gameplay implications.

3.7 Inference

To simulate the flow and creation of information more realistically in the NPCs, they can also deduce new Information from the ones they have. These deductions are based on rules that are globally defined and represent a form of cohesive causal understanding of the game world. To define such rules, there is the `InferenceRule` class. It consists of a `BoolExpression` object that defines the rule and a list of `Information` objects that define the consequences or outcomes if the rule applies.
The `BoolExpression` object holds `Information` objects and is chained together with other `BoolExpression` objects by logical operators to create the rule. Listing 3.2 defines the following rule for any agent X:

$$\mathit{isArmed}(X) \land \mathit{isEnemy}(X) \Rightarrow \mathit{isDangerous}(X)$$

```
var rule = new InferenceRule(new BoolExpression(new Information(
    _player, WorldAdjectives[Adjectives.Armed])).And(new
    BoolExpression(new Information(_player, WorldAdjectives[
    Adjectives.Enemy]))));

rule.Consequences = new List<Information> { new Information(_player,
     WorldAdjectives[Adjectives.Dangerous]) };
```

Listing 3.2. Creating an inference rule with consequences

The variable `_player` in the statement is just a placeholder and is replaced with different subjects during the evaluation of the rule. For the scope of this thesis, it made only sense to substitute for information subjects since the *Information Model* is structured in such a way that the information statements say something about an information subject.
In order to evaluate such a rule, every agent has an *Inference Engine* that allows them to do so. The *Inference Engine* sees the agent's stable memories as its knowledge base. It goes through the rules and checks if the knowledge base of the agent satisfies the rule. If so, all the pieces of information that are defined in a rules' list of consequences are added to the stable memory of the agent(Listing 3.3). This is repeated until no new information is added since any new information could trigger another rule.
The `SatisfiesKnowledgeBase` function recursively checks if a rule's *Boolean Expression* applies to the agent's memories. An information's subject is selected as a candidate if the information matches the information in the *Boolean Expression* in all components except for the subject.

```
public void Evaluate(){
   foreach (InferenceRule rule in Rules){
      List<InformationSubject> candidates = SatisfiesKnowledgeBase(
           rule.Expression, new List<InformationSubject>());

      if (!rule.AppliesToSelf)
         candidates.Remove(KnowledgeBase.Owner.InformationSubject);

      if (candidates.Any()){
         candidates.ForEach(c =>{
            rule.Consequences.ForEach(r =>{
               r = new Information(c, r);
               KnowledgeBase.TryAddNewInformation(r, KnowledgeBase.
                    Owner);
            });
         });
         Evaluate();
      }
   }
}
```

Listing 3.3. Evaluating inference rules

This way, agents can "reflect" on what they know and deduce new information.

3.8 Heuristic

Each agent is able to subjectively evaluate the believability of pieces of information it has stored in its memory. A metric is needed to measure the believability of information. I approached this via the "interest" of information. This means how unique a piece of information is or how probable is it that a piece of given information could be authentic. This depends on the "interest" of the involved information components. The information "*Farmer* `HAS` *milk*" is much more probable or uninteresting" than the information "*The king* `HAS` *the magical sword*". So by evaluating the components of the information and their relation to one another, a value for the "interestingness" of information is calculated. I call this value *Information Heuristic*.
The calculations for the *Information Heuristic* are, of course, slightly different depending on the types of the involved information components. The general idea is that the more "interesting" a piece of information is, the higher its *Information Heuristic* is. To make this value subjective to the agent, values that depend on the agent's experience or relation to involved agents are taken into consideration.
The basic formula is as follows:

$$h' = h\; h^v\; b\; n_i \sum_{a \in K} trust(a) \tag{3.1}$$

with $K := \{A \cap I\}$, where A is the set of known associates of the agent, and I is the set of agents this information was previously received from.
The value h^v is the part of the heuristic that is dependent on the verb and the information components, b is the previous believability of the information, and n_i is the number of times this information has already been received.
In other words, an information's heuristic is the product of itself with the component-dependent heuristic, the believability, the number of times it has been received, and the sum of the trust values of known agents this information has been received from.
In other words, an information's heuristic is the product of itself with the component-dependent heuristic, the believability, the number of times it has been received, and the sum of the trust values of known agents this information has been received from.
Now the component-dependent heuristic is another heavy calculation function. It differentiates between the verb and then, in each case, calculates the value differently depending on the components. If another agent is involved, for example, the distance or degrees of separation between the bespoke agent and the receiver of the information is taken into account. The calculation is done using a *breadth-first search*. Breadth-first search is an algorithm to search finite graphs. One of its advantages over for example depth-first search, is that it can be used to find the shortest path between two nodes u and v, with the path length measured in the number of edges between the nodes [10]. This should reflect cases where information is received like "I heard from a friend of a friend...". The longer the distance, the less trustworthy the information. If the subject is a known associate, then the trust the agent has towards the subject is also considered. The number of other pieces of information the receiver has concerning the subject is also used. This should reflect a familiarity with the subject.
If the subject of the information is an item, then not only is the uniqueness of the item in the world relevant, but also how unique the item is for the agent. This takes into account whether the agent owns an item of the same type and how many other pieces of information it has that refer to an item of the same type. This again is to simulate how familiar the agent is with the bespoke item.

So now there is a value for every information an agent has that says how "interesting" or "probable" that information is. To calculate the believability of information, this heuristic is now put into context with every other information an agent has. I define the *distance* between two pieces of information as the absolute difference between their heuristic values.
For the believability b of a piece of information this results in:

$$b = \frac{\sum_{i,j \in M, i \neq j} h(i) - h(j)}{n} \tag{3.2}$$

In this formula, n is the size of the stable memory, and M is the stable memory itself. The function h is the heuristic function. This means the believability of information is the average distance to every other information in the stable memory.
Since this is calculated for every information in the stable memory of every agent, this operation needs a lot of processing power. I implemented the MapReduce programming model to enable the distribution of these calculations. The prototype does not leverage the distribution itself since this was too big a task for the scope of the thesis, but the implementation is prepared to be used with little additional work (Listing 3.4). The *Map* and *Reduce* steps are marked with comments in the listing.

```
public void UpdateBelievability(){
   List<InformationContext> data = GetAllMemories();

   // Map
   var distances = new List<List<float>>();

   for (int i = 0; i < data.Count; i++){
      distances.Add(new List<float>());
      for (int k = 0; k < data.Count; k++){
         Information i1 = data[i].Information;
         Information i2 = data[k].Information;
         if (!i1.Equals(i2))
            distances[i].Add(GetInformationDistance(i1, i2));
      }
   }

   // Reduce
   for (int i = 0; i < distances.Count; i++){
      if (ContainsStableInformation(data[i].Information) && distances[i
          ].Any())
         GetStableMemory(data[i].Information).Believability = distances
             [i].Average();
      else if (ContainsSpeculativeInformation(data[i].Information) &&
          distances[i].Any())
         GetSpeculativeMemory(data[i].Information).Believability =
             distances[i].Average();
   }
}
```

Listing 3.4. The MapReduce implementation for calculating believability

Just how expensive the call to the `UpdateBelievability` function is can be measured with Unity's *Profiler*. It is a tool that allows exact time measurements of the different function calls in the Unity engine (Figure 3.12). The blue spikes in Figure 3.13 show spikes

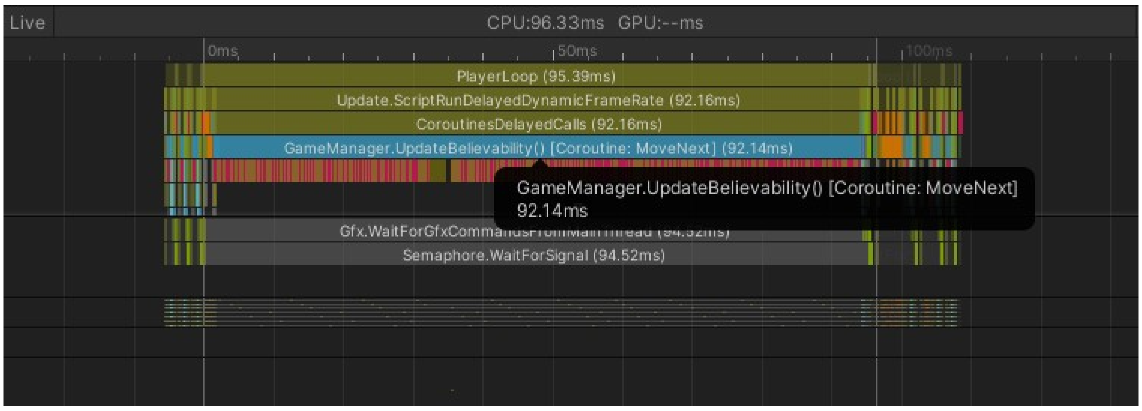

Live
CPU:96.33ms GPU:--ms
0ms
50ms
100ms
PlayerLoop (95.39ms)
Update.ScriptRunDelayedDynamicFrameRate (92.16ms)
CoroutinesDelayedCalls (92.16ms)
GameManager.UpdateBelievability() [Coroutine: MoveNext] (92.14ms)
GameManager.UpdateBelievability() [Coroutine: MoveNext]
92.14ms
Semaphore.WaitForSignal (94.52ms)

Figure 3.12: Frame data of the UpdateBelievability function call.

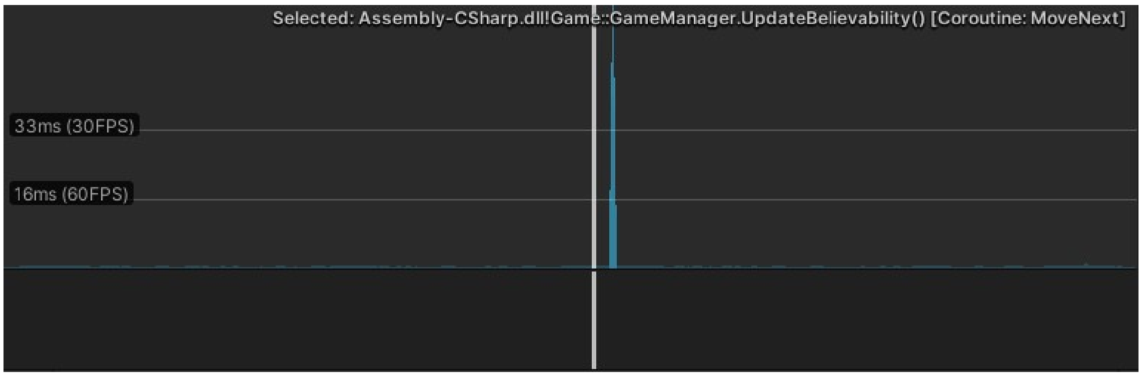

Selected: Assembly-CSharp.dll!Game::GameManager.UpdateBelievability() [Coroutine: MoveNext]
33ms (30FPS)
16ms (60FPS)

Figure 3.13: UpdateBelievability function call time.

in the calculation times when the function is called. The frame data was captured about two minutes after the game started. By this time, all the agents had moved around for a while and learned new information. The function call takes 92ms. Almost a second of calculation time is an immense performance issue. The function call is also perceivable in the game as there is a noticeable frame drop when the believability is calculated. This shows that distributing the call by truly applying the MapReduce model and using multiple nodes to calculate the believability would be a worthwhile adaption of the prototype. The amount of time this takes is not that important if it does not halt the game. A a simulation of the human process of "reflecting on one's thoughts" is something that could actually profit from a little waiting time.

3.9 Consensus

The prototype game differs from traditional multi-agent or even distributed systems in some significant ways. There are different requirements for it, and to a degree, it is even expected to behave differently.
For one, the system described in this thesis is not required to be concurrent from a conceptual point of view. It is not necessary that it responds as instantaneously as possible. As mentioned before, in fact, it might serve the simulation if specific processes take some time because they should simulate human thought processes. Concurrency is essential nonetheless to prevent performance issues when calling CPU-intensive functions.

Availability is a different issue than for conventional distributed systems since the game is not accessed remotely. Of course, if parts of the system were to be distributed, accessibility for those would need to be as high as possible.
Fault tolerance is, of course, necessary for any software system. In the specific case of this game mechanic, it is again desired behavior to allow for some loopholes that can be exploited by the agents. The game mechanic enables the player to do so by letting them introduce arbitrary and potentially factual wrong information into the system.
Sending messages in the game takes a relatively long amount of time. This is deliberate to simulate human "word of mouth" behavior. Humans cannot instantly transfer thoughts. This is why reaching actual consensus is such a problem for this system and not even the end goal.
To consider the consensus problem for the game is still a practical examination of how communication works between intelligent agents. It is true that the system should be exploitable by the player, but at the same time, the goal of the simulation is to mimic human communication, which means some form of authentication should be present as humans will not accept anything they hear.
The game mechanics achieves this by applying the *proof of personhood* method for permissionless consensus. Although the applications for actual consensus-reliant systems should be designed to allow *only* humans to validate their system changes, the principal idea is beneficial even for the simulation of human behavior.
In the prototype game, agents are verified by their peers based on their *trust* value. The trust value is dependent on the believability of the pieces of information an agent sends out(Listing 3.5). If a piece of information is not very believable, trust decreases and vice versa.

```
float believability = _stableMemory[idx].Believability;
if(Owner.Equals(sender))
   Owner.Acquaintances[sender] = 1.0f;
else
   Owner.Acquaintances[sender] += believability / 10.0f;
```

Listing 3.5. Calculating trust for proof of personhood

For a received information at position `idx` of the list `_stableMemory`, the trust value is defined as:

$$t' = t + \frac{believability}{10} \tag{3.3}$$

A small fraction of the believability is added to the former trust value t. Since believability can be negative and positive, and trust can likewise decrease or increase. The fraction enforces more minor changes and not too drastic fluctuations in the trust values. If the sender of the information is the receiver, then the trust value is set to 1 because every agent trusts itself completely.
The trust value of an agent then also influences the believability of the information. Of course, a more complex function for trust could be defined that takes strong relationships into account. For example, it could be more challenging for very trustworthy agents to lose trust and harder for untrustworthy ones to gain it. For the scope of this thesis, the present calculation is sufficient.
By influencing the result when receiving information, the trust value creates a metric for authenticating an agent when interacting with another agent. The authentication present in the prototype is only on a "one on one" basis that takes place at every information ex-

change. Agents do not authenticate themselves globally in the system, which is another big difference from conventional applications of consensus methods.
Back to the matter of consensus itself, I mentioned earlier that due to the time it takes for information to spread, it is unlikely that actual consensus among the agents is reached. That behavior, however, is beneficial to the simulation.
Another metric that is considered for the believability of information is how often it was received and how trustworthy those that it was received from are. This is actually a measurement that takes more than one agent into account and creates a weaker form of consensus while not communicating with many other agents. If information is received more often, agents assume that the probability of it being true is higher. Again this is an anchor to exploit the system to scatter wrong information and create lies that are advantageous to their playstyle. The quests in the prototype are designed to allow for different kinds of such playstyles.

3.10 Quests

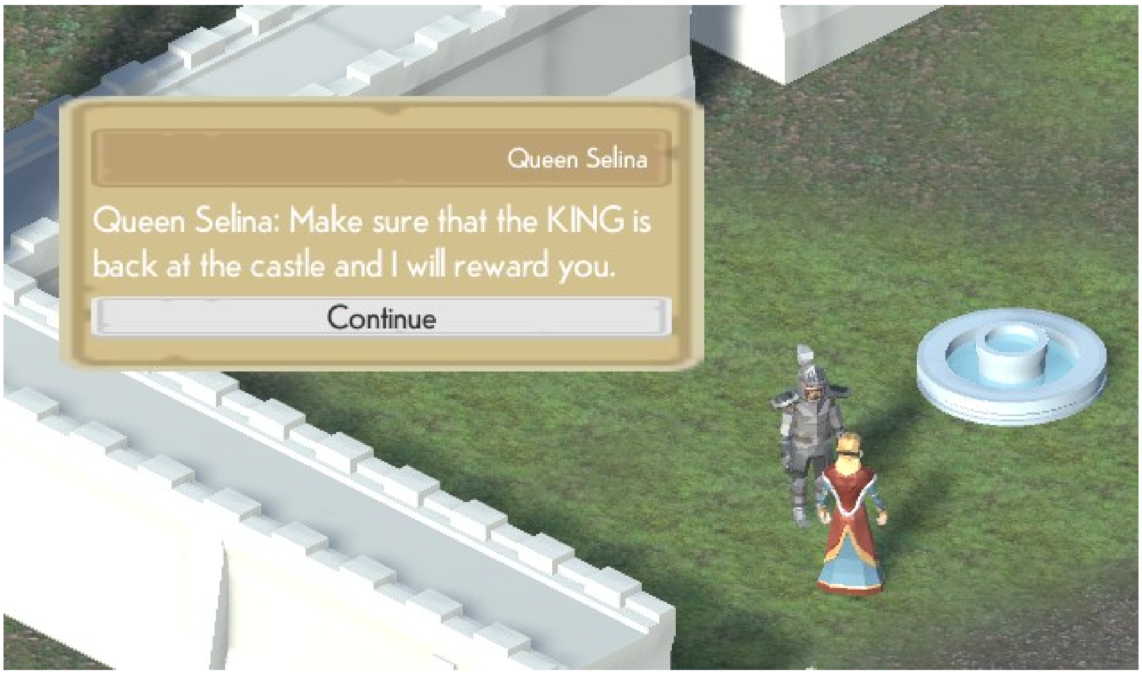
To give the player an objective and an incentive to take action in this game, quests can be created. They have associated lists of information pieces and inference rules. For a quest to be completed, either the quest giver has to know about the required quest information, or the required inference rules have to satisfy the knowledge base of the quest giver.
That makes the quest state entirely subjective to the quest giver. Rather than being managed globally by a *Game Manager*, the quest is solely dependent on the *Information Manager* of the quest giver.
In the scope of the prototype game, I have created one quest that is available to complete. At the start of the game, when talking to the queen in the castle, she will task the player with returning the king to the castle (Figure 3.14). This requirement can be created with one inference rule (Listing 3.6).

```
var rule = new InferenceRule(new BoolExpression(new Information(
    _player, WorldAdjectives[Adjectives.King]))).And(new
    BoolExpression(new Information(_player, GameObject.Find("Castle"
    ).GetComponent<Location>())));

Quest q = new Quest(GameObject.Find("Queen").GetComponent<NPC>());
q.GoalRules.Add(rule);
```

Listing 3.6. Creating an inference rule for a quest

The rules simply define two requirements: Any character who *is* the king and who is *at* the castle, at the same time, will satisfy this rule. This creates *objective-oriented* quests following Aarseth's classification [1]. There are only goal requirements that define the quest. All the steps to achieve this goal are up to the player. Combined with the dialogue system, however, this system also allows creating more controlled quests. Dialogue can play out differently based on variables that can be set depending on what the player does in the world. Additional inference rules could be added that cover different outcomes for the quest. Quest giver characters could then react differently to those outcomes. The quest system is flexible to allow varied gameplay approaches through defining only goal requirements but allowing developers to take a more narrated approach by providing additional story content for specific combinations of outcomes.



Queen Selina: Make sure that the KING is back at the castle and I will reward you.

Figure 3.14: A quest is presented to the player.

3.11 Narrative

With the presented game mechanic, there are numerous narrative implications that need to be addressed. I have described *how* the quests provided in the prototype game are purely defined through their goals. They do not specify how exactly those goals have to be reached, although they could do so by adding more restrictions to the goal.
The intention of the game mechanic is that precisely through this careful definition of goals, player agency is increased because they are less restricted in their choice of actions to accomplish a given goal. By freeing up the options for players, the narrative structure of a quest can change drastically.
Going back to the quest present in the prototype, I described two possible approaches that are very different not only in the actions necessary to complete the quest but also in the stories that are told.
The first approach is more aligned with conventional role-playing quest lines. The player takes the role of the knight who is tasked to bring the king back to the castle safely. In the village near the castle, the player can talk to NPCs to learn pieces of information about the world. They can also ask specifically for the location of the king. The king is guarded by the dragon at the clearing in the northwest area of the kingdom. If the player manages to learn the king's location from one of the NPCs, they can travel to that location and will find the dragon and the king. The king will tell the player that the dragon will not let him go, and the player needs a weapon in order to defeat the dragon. Many NPCs carry a sword, but they will not simply give it to the player. There is, however, a sword in the stone circle north of the village. Players can either learn of its existence from other NPCs or just stumble across it while exploring. Interacting with the sword will allow the player to carry it. Now the player can go back to the clearing to defeat the dragon. Upon doing so, the king will start walking back to the castle. He will tell the player that he will inform the queen of the player's deeds once he arrives. The player can then go back to the castle

and wait for the king's return. When the queen meets him, she will learn that the king is back at the castle, and the quest will be finished.
This storyline is familiar and straightforward. It has a clear beginning, middle, and end. The king has been kidnapped by a dragon, and a brave knight sets out to rescue him. To defeat the dragon, the knight needs to find a mighty sword to defeat the beast. The knight wanders the kingdom in search of this weapon until he reaches it. Now he can slay the dragon and thus save the king who reclaims his rightful throne.
The structure resembles the hero's journey as defined by Campbell [12]. The hero experiences adventures, faces challenges and their world undergoes a change. In the presented story the challenges are defeating the dragon and obtaining a powerful weapon. The changed world is the recognition of their valiant actions and the return to a stable kingdom under the king's rule.
In the prototype, all the storytelling is done on a plot level, meaning through the events that are transpiring. Character-driven storytelling is much more complex to simulate without strongly scripting the dialogue. One step towards more character-driven stories would be by including the previously mentioned emotion engines to enable NPCs to simulate more human reactions to events. However, to articulate the characters' feelings, the dialogue system still would need to be fundamentally reworked and or extended, which exceeds the scope of this thesis.
The second approach to complete the quest I presented is to convince enough people of the fact that the knight is the king. If an NPC with a high enough trust value tells this to the queen, she will believe that the knight is now the king, and since they are present at the castle, she will also learn the king is at the castle, which satisfies the quest objectives. This, albeit being a much shorter storyline, has more complex implications since it is not so aligned with narrative conventions players have learned to expect. Here we have a story of a cunning opportunist who wants to seize the power vacuum left by the absent king. The knight travels the kingdom and pronounces themself the new king. Through their sneaky deception of the people in the kingdom, word reaches the queen that the knight is the king. Since this is brought to her by a trusted soldier, she is led to believe that it must be the truth. So when the knight returns to her, she greets them as the new regent.
In that story, the player takes on the role of the villain, or at least an anti-hero, since their methods are not aligned with traditionally heroic values. It could be that the actual king is a tyrannical despot, which would make resorting to such measures more acceptable, or the queen could be part of the ruse to replace the king. This story is not structured differently than the first one and adds morally gray undertones to the player characters.
The two storylines I have presented are both possible from the simple setup of the quest that is part of the prototype. The actions the player takes in the game are similar and mainly rely on the exchange of information, which is the heart of the game mechanic. Nevertheless, even though the quest is completed in both cases with the satisfaction of the goal conditions, the narrative implications of the two variants are fundamentally different both in their story structure and even in the story genre. Where the first story is a classic fantasy adventure story, the second is more a story of political deception.
Of course, all these potential elements of narrative framing are not part of the prototype. The present dialogue system would allow for reactions depending on the player's actions because the Yarn Spinner dialogue allows for the use of variables that can be tied to the gameplay events. Nevertheless, the use of game world information creates varied possibilities for completing objective-oriented quests that offer exciting story explanations.

4 Comparison

In order to properly evaluate the novelty and applicability of the presented game mechanic, it needs to be compared to different available games that employ similar game mechanics. This will help classify the game mechanic created during this thesis and how it holds up next to already released games.

4.1 Comparison criteria

There are many games that contain some form of dynamic storytelling, procedural narrative content, or emergent storytelling. Across the video game industry, those terms are not clearly defined and are sometimes used interchangeably or in a conflicting way. For that reason, while researching games where a comparison would be worthwhile, I decided on a set of criteria that I will use to differentiate the different games. The absence or presence of these criteria does, however, say nothing about the quality of a game, just whether it features a specific manifestation of dynamic storytelling.

4.1.1 Reactiveness

With these criteria, I evaluate whether a game changes depending on the actions of the player. It could be that narrative content is generated or remixed on every new game but does not actually react to anything the player does. This criterion should show, however, if the game's narrative content does change as a consequence of player actions.

4.1.2 Narrative Awareness

Another important aspect is whether the change in narrative content is something that is acknowledged and reflected in the game or is the narrative change purely emergent and has to be interpreted by the player. Is the game aware of its change in narrative, or does it behave in such a way that the change is only apparent outside of it?

4.1.3 Scripted Narrative

When it comes to the narrative content itself, it is essential to consider how much of it is generated by game logic and how much of it is scripted beforehand by developers. This would include small bits of scripted narrative content that are reordered to create a new narrative.

4.1.4 Replayability

Finally, is the narrative generated in such a way that it would change when starting a new game, or will it be the same when the player does the same things? In other words, is the narrative generation deterministic or not?

4.2 Games to compare

To classify my own prototype in these criteria, I have selected several games that feature generated narrative content. Among the selected games are titles that have been available for several years, as well as games that were released very recently as of the writing of this thesis. I tried to select games that would all be considered in the area of *generated narrative* or *dynamic storytelling* but are very distinct and different examples of such narrative features. In the following, I will shortly describe the selected games and how they feature generated narrative content.

4.2.1 Road 96

In *Road 96* [23], the player takes on the role of multiple runaway teenagers who want to make their way across the border of a fictional country before an upcoming election might hinder them from leaving the country. The game is presented as a series of encounters with people who also find themselves on the road. It has only a small amount of gameplay systems like an energy bar or the tracking of money.
The heart of *Road 96* is its multiple-choice dialogue segments with the different characters. At every new start of the game, the player will be put in the shoes of a different hitchhiker and meet different people from a recurring cast of characters. Environmental details are changed every time, and the dialogue options are dependent on the characters and whom they meet. *Road 96* is a very complex choice tree that the player traverses as characters encounter each other. The choices players make during the game sessions will influence the ending of the individual character. All possible combinations of characters are, however, previously written and finite, yet exhaustive.

4.2.2 Crusader Kings III

Paradox's *Crusader Kings III* [75] is a political strategy and simulation game set between 867 and 1453 across Europe, Asia, and Central Africa. The player controls the heads of their dynasty across many generations who have character traits and skills that develop over the game. Political choices such as trading, warfare, and diplomatic interactions change the political landscape and create complex game states.
The game has very complex, interlocking systems, including several government types, a genetic system for passing down character traits to descendants, a social system for interacting with other characters, and many more. This leads to incredibly varied and complex narrative scenarios because the different systems interact with each other.

4.2.3 Divinity: Original Sin II - Game Master Mode

Part of *Divinity: Original Sin II* [57] is a game mode that is still quite unique in the video game landscape - the *Game Master Mode*. The base game *Divinity: Original Sin 2* is a critically acclaimed fantasy role-playing game that is similar to genre contemporaries that are based on tabletop games like *Dungeons and Dragons*. Players create a character with a set of skills and stats that determine what they can do in a game, how they fight, and how they interact with other characters in the game.
The addition of the *Game Master Mode* now allows players to create content within the context of *Divinity: Original Sin 2*. It is designed to be played as an asymmetrical multiplayer game where one player takes on the role of the *Game Master*. This role opens up the game engine and lets the player access a content creation toolkit. They then can create narrative scenes, place objects and characters in the 3D environments and prepare encounters, quests, and interactions. When players join such a scenario, they can then play through it interactively and more in the style of a conventional tabletop experience. The players control their characters and use game mechanics and systems from *Divinity: Original Sin 2* but experience a story that is entirely user-created. There are additional tools available to the *Game Master* that enable player groups to react to situations that exceed the limitations of the base game. Additional rules can be created or brought into the game from other role-playing game systems.
This means that the narrative content is not technically *generated* by the game, but it is highly adaptive and more variable because it introduces an actual human narrator.

4.2.4 Darkest Dungeon

When released in 2016, *Darkest Dungeon* [79] was highly praised for its rich atmosphere and immersive setting. In the role-playing game, the player leads a party of randomly generated adventurer characters into procedurally generated dungeons. There, they encounter threats and obstacles in the form of enemies and negative character modifiers like mental disorders and other emotional traumas. The game focuses on the psychological toll adventures take on the protagonists and presents a sinister and dark fantasy world.
Although there is an overarching plot in the game, it is only dealt out in tiny bits in order to tie the many dungeon expeditions together. The expeditions themselves are presented as "mini-stories" in which characters venture into the unknown and encounter terrible dangers that leave them scarred forever. A narrator's voice adds atmospheric one-liners that underline the characters' struggles and weaves a narrative throughline into each expedition.

4.2.5 Unexplored 2: The Wayfarer's Legacy

A very recent title is *Unexplored 2: The Wayfarer's Legacy* [61]. It is another role-playing game that uses its interacting gameplay systems to create emergent narrative content. A created character is tasked with destroying a magical object that could allow an evil force to conquer the world. Players have to explore, fight, and talk their way through different challenges. Player actions however, do not happen in a vacuum and changes are persistent even if a character dies. The world does not reset and randomize itself. Each death moves the clock forward a few years. During those intervening years, the antagonist party's influence expands and makes the journey more dangerous. Alliances

between clans strengthen or dissolve, and new quests replace old ones. Although the map stays the same, the layout, enemies, and challenges found in each individual area change. When it makes sense, though, certain things persist in between games. For example, unique objects gained in one game and then lost because of death could be recovered by the next character at that exact location or might have been taken away by an NPC.
There is not much of a main story present in *Unexplored 2*, but all of these complicated systems interlock in ways that organically create a new one with each character. Even the skill check system tells little stories. That is frequently used to determine everything from dialogue resolution to solving stat-based puzzles where success is partially influenced by the character's skills.

4.2.6 Prototype Game

The prototype game developed as part of this thesis has been described in great detail already. I just want to recap its narrative properties in the context of this classification.
The prototype game is modeled as a multi-agent system that allows the exchange, retrieval, and introduction of information with other agents. There is no built-in AI that tries to create storytelling behavior, but the designers can create quests that are defined through rules that need to be satisfied. The goal of the prototype is to create an emergent narrative through the interaction of its systems.

4.3 Results

I have analyzed several games through the lens of the described classification criteria. The selected games all feature mechanics that fall under the umbrella term of *dynamic storytelling* but apply the creation of narrative content differently.
The final result of the classification shows that the games all behave in different ways when replayed multiple times. This is not yet true for the prototype game since the game does not yet properly react to the player (Table 4.1). The generated narrative is purely emergent.
All the considered games except for *Road 96* and the prototype game actually react to player actions. Games like *Road 96* follow a more "choose your own adventure" approach where all options are known upfront and are then mixed and matched to create new variations.
Three of the analyzed titles, *Road 96*, *Crusader Kings III*, and the *Game Master Mode* in *Divinity: Original Sin 2*, have capabilities to generate narrative content that follows a dramatic structure. They do so differently, though by reordering encounters in a more dramatic fashion, using the game AI to let characters behave in specific ways, or by introducing a human narrator to steer the storyline.
It is worth noting that all games except for *Crusader Kings III* contain some form of existing storyline or narrative frame in which the game takes place. Only *Crusader Kings III* lets its gameplay systems alone in combinations with the player actions dictate the unfolding, fully emergent story.

Game	**Reactiveness**	**Narrative Awareness**	**Scripted Narrative**	**Replayability**
Road 96	No	Yes	Yes	Yes
Crusader Kings III	Yes	Yes	No	Yes
Divinity: Original Sin 2 - Game Master Mode	Yes	Yes	No	Yes
Darkest Dungeon	Yes	No	Yes	Yes
Unexplored II: The Wayfarer's Legacy	Yes	No	Yes	Yes
Prototype Game	No	No	Yes	No

Table 4.1: Results of game comparison

5 Discussion

This chapter puts the results of the prototype game into perspective. I will discuss which parts of the declared goals have been achieved and which ones need further work. Furthermore, the challenges for adapting the presented approach in the given scope are explained. Finally, potential future research and work areas are discussed.

5.1 Achievements

The goal of this thesis is to create a novel game mechanic that uses the exchange of game state information to support narrative content creation while still allowing for narrative framing from designers.
I achieved this by simulating the information exchange by modeling a multi-agent system of NPCs. The resulting prototype shows this core mechanic quite proficiently in regards to the simulation of information flow. The agents move around the game world and exchange information that they learn with one another. They tend to share more "interesting" information. The quality of "interest" of information is calculated by taking the source of information into account as well as the probability of the information being true. This creates a much-needed context for the information exchange that changes for each agent as they meet more other agents and learn new information.
There is also a social aspect to the exchange of information. Each character has a set of other characters that form their "circle of influence". This is a group of characters who will be trusted more than other agents, which means that information coming from them will be seen as more believable. This way, friends, family, or other influential relationships are modeled in the system as well.
When it comes to the narrative capabilities of the prototype, I have described in previous chapters how it allows for different playstyles that result in vastly different narrative structures. By carefully crafting quests in an objective-oriented manner, it is possible to leave the methods of completing a quest quite open to the player.
This means that the game mechanic can achieve the goal of supporting or even creating storylines that are not previously defined by a designer but originate in the player's actions in the decision during the game.
From a technological point of view, the prototype uses techniques from several areas of computer science and applies them in order to support the desired outcome. It is even prepared in such a way that it could distribute processing-intensive calculations to other processes in order to optimize performance. That is something that not many games use yet since it would require a separate infrastructure of support computers that need to be available in order to play the game.
Other methods used in the prototype are consensus protocols, state machine modeling, and inference engines. The prototype takes aspects or concepts from each of them and uses them to support the game mechanic. The result is a prototype that is accessible to somewhat experienced players of video games that is capable of showing the game mechanic in a robust and consistent way.

5.2 Shortcomings

Not all desired features or capabilities found their way into the prototype game due to the scope of the thesis or other challenges. In this section, I want to describe the aspects that still require additional work.

5.2.1 NPC Reactions

Maybe the most crucial missing feature of the prototype is the actual reactions of agents to learning information. NPCs do adapt their internal state and evaluate other characters and pieces of information differently with the information they have and learn, but they do not adapt their behavior in any significant way. This could genuinely elevate the narrative capabilities of the prototype to the next level because it would allow even more dynamic behavior and situations that carry narrative implications. It would have required extensive additions to the *Information Model* and to the agent AI to do so, and sadly this was not possible in the scope of this thesis.

5.2.2 Adaption of the NPC Routine

One approach to introduce reactions to the agent behavior would be to implement insertions or other manipulations to an NPC's routine. The routine is the list of behaviors an agent executes in sequence in the game. Allowing changes to that list based on learned information would introduce more dynamic behavior as well while allowing the NPC to return to their original stable routine after an inserted behavior has been executed.

5.2.3 NPC Actions

Generally speaking, in order for NPCs to more dynamically react to changes in the game state or to new information, they need to be able to perform more varied actions. The prototype is implemented in a way to create and add new behaviors for NPCs easily, but of course, the present version would have profited from more available behaviors. Examples could include fighting, trading, or performing professional jobs. Modeling complex agents was, however, not the focus of this thesis which is why other behaviors were not extensively developed.

5.2.4 Conversation

One specific NPC behavior that would instantly make the agents seem more active and goal-driven as compared to the current state would be the ability to start a conversation on their part. With the already existing system to ask for or introduce specific information, NPCs could walk up to another player and ask them for a specific piece of information. This could also be used to make NPCs able to lie to others by telling them semantically correct but factually wrong information actively. Tied to a more capable NPC AI, this would make them into more active participants in the game.
Additionally to NPCs not being able to initiate a conversation, the dialogue system right now is also fundamental. Information is presented textually in a very raw form that is usually grammatically incorrect yet understandable. Additional work on the dialogue

system using, for example, generative grammars could increase both readability and variation in the dialogue lines.

5.2.5 Overall Content

Finally, although the prototype game world is aesthetically pleasing and large enough for exploration to take place, it lacks interactable content. There are many agents walking around and going after their business which was the focus of the thesis, but when it comes to items to interact with and interactions to perform, the prototype is lacking. It was again the main focus to show the capabilities of the game mechanic in the scope of this thesis, and adding more varied content is something I would gladly explore in the future.

5.3 Outlook

In addition to the described areas that improve and extend existing systems, there are several areas that would require extensive reworks and additions that would make the game more novel and more innovative.

5.3.1 NPC Learning

First on this list would be the ability to learn for NPCs. By that, I mean utilizing machine learning or reinforcement learning techniques to allow them to evaluate information more accurately. The Unity engine contains so-called *Machine Learning Agents* to provide just such capabilities. They use reinforcement learning and intimidation learning to train the agents that can then be used as agents in a game.
Adding complexity to the *Information Model* would also enable significant improvements for the decision-making processes of the NPCs. I mentioned *Emotion Engines* in Section 2.8.1, which would be a worthwhile addition to my own *Information Model*. Extending the social awareness of the NPCs could also yield more character-driven narrative structures when they are interacting.

5.3.2 Towards Intelligent Agents

In order to leverage the more complex decision-making and social capabilities of NPCs, a more complex AI would also be a prospective field of further research. Since the quests in the prototype game are already defined as objective-oriented, work could be done towards allowing the NPCs to follow those goals as well, or even goals of their own. They could be equipped with their own motivations and priorities that change then depending on what they learn or deduce. Truly intelligent agents would very probably yield to unexpected and interesting narrative outcomes and situations and make the game indeed "come alive".
By then, the game would be best described as a *social simulation* that uses quests to create agency for the player. However, with all the systems interacting on such a deep level, players could also work towards goals of their own design and see how far they can bend the limitations of the simulation.

5.3.3 Narrative Awareness

Finally, to really add to the narrative novelty of this game mechanic, it would be an auspicious direction to do research into the area of narrative awareness of the game. By that, I mean to design the AI of the agents and maybe an overseeing game logic entity to steer the parts of the game towards more dramatic decisions. For that to work, stories would have to be formalized in such a way that the game logic could evaluate the current game state and then decide what would need to happen to create a satisfying dramatic situation. This could then be translated into the behavior of single agents who would change their behavior in order to satisfy the dramatic needs of the game. This would go in the direction of creating a virtual *Game Master* that is common in tabletop role-playing games. An entity that controls all the NPCs and events in the game so that an interesting story unfolds while still being able to react to player actions and incorporate them into its own decision-making process.

6 Conclusion

In the world of consumer media, creators are constantly looking for new ways to tell stories. Since video games are inherently interactive, they provide entirely new and innovative ways to experience and create stories. Dynamic storytelling is a relatively new area in game design and game development that focuses on creating satisfying narrative experiences without the need for extensive writing or the previous creation of a story. Dynamic storytelling is an umbrella term for different approaches to achieve this goal, such as artificial intelligence to drive the choices of NPCs, complex systems of interlocking game rules that create unexpected outcomes with narrative implications, or small bits of previously written narrative content that are rearranged and slightly changed to form new narratives. Using emerging technology is beneficial to exploring these new ways.
This thesis presents a novel game mechanic that supports dynamic storytelling by allowing NPCs to exchange game state information and allowing the player to retrieve and introduce pieces of information to the system. The developed prototype game applies techniques from multiple disciplines of computer science in order to implement the designed game mechanic. Concepts from *Emergence Games*, *Communication Theory*, *Multi-Agent Systems*, *State Machines*, *Consensus Protocols*, *MapReduce*, and *Inference Engines* are brought together to create systems that interact in a deep and complex way to allow the NPCs to make decisions and act on them. This approach shows potential that encourages the further development of this game mechanic.
I explain the theories and the research behind these techniques and provide the necessary information to understand the implementation of the prototype. This thesis touches on many of the topics that were part of my syllabus, both the Bachelor and the Master program of Games Engineering, and as such, provide a good throughline of the learned theories and concepts. I then give a detailed overview of the different components and systems that work together in order for the prototype to reflect the game mechanic. Sometimes I used only aspects of a technique or the basis of a concept to apply in the prototype, but the underlying thoughts are essential for the game mechanic once put together.
Finally, I compare the developed prototype game to other games that apply different methods of dynamic storytelling and classify those games according to important criteria of dynamic storytelling. The developed prototype provides exciting and varied narrative scenarios depending on the actions the player executes in the game. There are, however, several shortcomings that hinder the prototype from being more immersive in its narrative implications and more varied in its gameplay options. I suggest areas where further work is most promising and where the basic groundwork is already present in the prototype to be expanded.
To conclude, there is much untapped potential in the area of dynamic storytelling. This novel approach to leverage information exchange to a core game mechanic to support narrative gameplay is a further step into this area.

List of Figures

List of Tables

Listings

Bibliography

[1] Espen Aarseth. "From Hunt the Wumpus to EverQuest: Introduction to Quest Theory". In: *Entertainment Computing - ICEC 2005*. Springer Berlin Heidelberg, 2005, pp. 496–506. DOI: 10.1007/11558651_48.

[2] Porter H. Abbott. *The Cambridge introduction to narrative*. Cambridge University Press, 2020.

[3] R.B. Adler, G.R. Rodman, and A. DuPré. *Understanding Human Communication*. Oxford University Press, 2016. ISBN: 9780190297084. URL: https://books.google.de/books?id=YF1fvgAACAAJ.

[4] Aristotle. *Poetics*. ReadHowYouWant. com, 2006.

[5] Chris Baldick. *The concise Oxford dictionary of literary terms*. Oxford University Press, 1996.

[6] Marc G. Berman, John Jonides, and Richard L. Lewis. "In search of decay in verbal short-term memory." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 35.2 (Mar. 2009), pp. 317–333. DOI: 10.1037/a0014873.

[8] Maria Borge et al. "Proof-of-Personhood: Redemocratizing Permissionless Cryptocurrencies". In: *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, Apr. 2017. DOI: 10.1109/eurospw.2017.46.

[9] Michael K. Buckland. "Information as thing". In: *Journal of the American Society for Information Science* 42.5 (June 1991), pp. 351–360. DOI: 10.1002/(sici)1097-4571(199106)42:5<351::aid-asi5>3.0.co;2-3.

[10] Alan Bundy and Lincoln Wallen. "Breadth-First Search". In: *Catalogue of Artificial Intelligence Tools*. Springer Berlin Heidelberg, 1984, pp. 13–13. DOI: 10.1007/978-3-642-96868-6_25.

[11] David Callele, Eric Neufeld, and Kevin Schneider. "Emotional Requirements". In: *IEEE Software* 25.1 (Jan. 2008), pp. 43–45. DOI: 10.1109/ms.2008.5.

[12] Joseph Campbell. *The hero with a thousand faces*. Vol. 17. New World Library, 2008.

[13] Alphonse Chapanis. "Interactive human communication". In: *Scientific American* 232.3 (1975), pp. 36–46.

[14] Michael F. Connelly and Jean D. Clandinin. "Stories of Experience and Narrative Inquiry". In: *Educational Researcher* 19.5 (June 1990), pp. 2–14. DOI: 10.3102/0013189x019005002.

[15] Daniel Cook. *What are game mechanics?* https://lostgarden.home.blog/2006/10/24/what-are-game-mechanics/. Accessed on: 2021-09-23. Oct. 2006.

[16] Gus Cooney, Daniel T. Gilbert, and Timothy D. Wilson. "The Novelty Penalty". In: *Psychological Science* 28.3 (Jan. 2017), pp. 380–394. DOI: 10.1177/0956797616685870.

[17] George Coulouris. *Distributed systems : concepts and design*. Harlow, England New York: Addison-Wesley, 2001. ISBN: 9780201619188.

[18] Mihaly Csikszentmihalyi. *Flow: The psychology of optimal experience*. Vol. 1990. Harper & Row New York, 1990.

[19] Grzegorz Czajkowski et al. *Sorting Petabytes with MapReduce - The Next Episode*. `https://ai.googleblog.com/2011/09/sorting-petabytes-with-mapreduce-next.html`. Accessed on: 2021-09-26. Sept. 2011.

[20] Michele R. Davidson. "A phenomenological evaluation: using storytelling as a primary teaching method". In: *Nurse Education in Practice* 4.3 (Sept. 2004), pp. 184–189. DOI: `10.1016/s1471-5953(03)00043-x`.

[21] Marie Dealessandri. *Global console market reached record high of $53.9bn in 2020*. `https://www.gamesindustry.biz/articles/2021-03-11-console-market-reaches-record-high-of-usd53-9bn`. Accessed: 2021-09-13. Mar. 2021.

[22] Jeffrey Dean and Sanjay Ghemawat. "MapReduce". In: *Communications of the ACM* 51.1 (Jan. 2008), pp. 107–113. DOI: `10.1145/1327452.1327492`.

[24] F. Dignum et al. "Games and Agents: Designing Intelligent Gameplay". In: *International Journal of Computer Games Technology* 2009 (2009), pp. 1–18. DOI: `10.1155/2009/837095`.

[25] Elizabeth A. Doty. "Transforming Capabilities: Using Story for Knowledge Discovery & Community Development". In: *National Storytelling Network's Storytelling In Organizations (SIG)* (2003).

[26] John R. Douceur. "The Sybil Attack". In: *Peer-to-Peer Systems*. Springer Berlin Heidelberg, 2002, pp. 251–260. DOI: `10.1007/3-540-45748-8_24`.

[27] Christopher Dring. *Eidos Montreal: "We have to try new models for single-player games"*. `https://www.gamesindustry.biz/articles/2018-05-11-eidos-montreal-we-have-to-try-new-models-for-single-player-games`. Accessed: 2021-09-13.

[28] William H. Edwards. *Motor learning and control: From theory to practice*. Cengage Learning, 2010.

[30] Michael Eysenck. *Attention and arousal: Cognition and performance*. Springer Science & Business Media, 2012.

[31] Edward A. Feigenbaum, Avron Barr, and Paul R. Cohen. "The handbook of artificial intelligence". In: (1981).

[32] Owen J. Flanagan. *Consciousness reconsidered*. Vol. 250. MIT press Cambridge, MA, 1992.

[33] Luciano Floridi. *Information: A very short introduction*. OUP Oxford, 2010.

[34] Luciano Floridi. "Philosophical Conceptions of Information". In: *Formal Theories of Information*. Springer Berlin Heidelberg, 2009, pp. 13–53. DOI: `10.1007/978-3-642-00659-3_2`.

[35] Dom Ford. "Beyond the Wall: The Boundaries of the Neomedieval Town in Singleplayer Roleplaying Games." In: *DiGRA Conference*. 2019.

[36] Chris D. Frith and Uta Frith. "Social Cognition in Humans". In: *Current Biology* 17.16 (Aug. 2007), R724–R732. DOI: `10.1016/j.cub.2007.05.068`.

[37] Hidemaro Fujibayashi, Takuhiro Dohta, and Satoru Takizawa. "Breaking Conventions with The Legend of Zelda: Breath of the Wild". In: *Game Developers Conference, March*. Vol. 10. 2017.

[38] Arthur Gervais et al. "On the security and performance of proof of work blockchains". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[39] Jonathan Gottschall. *The storytelling animal: How stories make us human*. Houghton Mifflin Harcourt, 2012.

[40] David Harel. "Statecharts: a visual formalism for complex systems". In: *Science of Computer Programming* 8.3 (June 1987), pp. 231–274. DOI: 10.1016/0167-6423(87)90035-9.

[41] David Harel and Amir Pnueli. "On the development of reactive systems". In: *Logics and models of concurrent systems*. Springer, 1985, pp. 477–498.

[42] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1983.

[43] Dirk Helbing. "Agent-Based Modeling". In: *Understanding Complex Systems*. Springer Berlin Heidelberg, 2012, pp. 25–70. DOI: 10.1007/978-3-642-24004-1_2.

[44] Stefan Hoffmann. *Geschichte des Medienbegriffs*. Vol. 3. Felix Meiner Verlag, 2000.

[45] John H Holland. *Emergence: From chaos to order*. OUP Oxford, 2000.

[46] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. "Introduction to automata theory, languages, and computation". In: *Acm Sigact News* 32.1 (2001), pp. 60–65.

[47] Emiliano Ippoliti, ed. *Heuristic Reasoning*. Studies in Applied Philosophy, Epistemology and Rational Ethics. Springer, 2015. ISBN: 978-3-319-36222-9.

[49] D. A. Jameson. "Narrative Discourse and Management Action". In: *Journal of Business Communication* 38.4 (Oct. 2001), pp. 476–511. DOI: 10.1177/002194360103800404.

[50] Jesper Juul. "The Open and the Closed: Game of emergence and games of progression". In: *Computer Games and Digital Cultures Conference Proceedings*. Ed. by Frans Mäyrä. Tampere University Press, 2002, pp. 323–329.

[51] Geoffrey Keppel and Benton J. Underwood. "Proactive inhibition in short-term retention of single items". In: *Journal of Verbal Learning and Verbal Behavior* 1.3 (Oct. 1962), pp. 153–161. DOI: 10.1016/s0022-5371(62)80023-1.

[52] Patrick Klepek. *Internal Sony Docs Explain How "Activities" Became a Cornerstone for PS5*. https://www.vice.com/en/article/5dp34k/internal-sony-docs-explain-how-activities-became-a-cornerstone-for-ps5. Accessed: 2021-09-13. Nov. 2020.

[53] Rune Klevjer. "Enter the Avatar: The Phenomenology of Prosthetic Telepresence in Computer Games". In: *The Philosophy of Computer Games*. Springer Netherlands, 2012, pp. 17–38. DOI: 10.1007/978-94-007-4249-9_3.

[54] Sarit Kraus. "Negotiation and cooperation in multi-agent environments". In: *Artificial Intelligence* 94.1-2 (July 1997), pp. 79–97. DOI: 10.1016/s0004-3702(97)00025-8.

[55] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* (July 1982), pp. 382–401. URL: https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/.

[56] Petri Lankoski and Satu Heliö. "Approaches to Computer Game Design–Characters and Conflict". In: *CGDC Confrence Proceedings*. 2002, pp. 6–8.

[58] Elaine J. Lawless. *Women escaping violence: Empowerment through narrative*. University of Missouri Press, 2001.

[59] Stephen C. Levinson. "On the human 'interactional engine'". In: (2006).

[60] Stephen W. Littlejohn and Karen A. Foss. *Theories of human communication*. Waveland press, 2010.

[62] Charles M. Macal and Michael J. North. "Agent-based modeling and simulation". In: *Proceedings of the 2009 Winter Simulation Conference (WSC)*. IEEE, Dec. 2009. DOI: `10.1109/wsc.2009.5429318`.

[63] Carlos Marín-Lora et al. "A game engine to make games as multi-agent systems". In: *Advances in Engineering Software* 140 (Feb. 2020), p. 102732. DOI: `10.1016/j.advengsoft.2019.102732`.

[64] Arthur W. Melton and W. J. von Lackum. "Retroactive and Proactive Inhibition in Retention: Evidence for a Two-Factor Theory of Retroactive Inhibition". In: *The American Journal of Psychology* 54.2 (Apr. 1941), p. 157. DOI: `10.2307/1416789`.

[65] Ibrahim Asif Mirza. "Development of an Emotion Engine for Games". MA thesis. Technical University of Munich, Apr. 2021.

[67] Temba Munsaka. *Communication is Complex. Definitions, Types and Problems*. GRIN Verlag, 2014.

[68] Muaz Niazi and Amir Hussain. "Agent-based computing from multi-agent systems to agent-based models: a visual survey". In: *Scientometrics* 89.2 (Aug. 2011), pp. 479–499. DOI: `10.1007/s11192-011-0468-9`.

[71] Wardrip-Fruin Noah et al. "Agency Reconsidered". In: *DiGRA &#3909 - Proceedings of the 2009 DiGRA International Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory*. Brunel University, Sept. 2009. ISBN: ISSN 2342-9666. URL: `%5Curl%7Bhttp://www.digra.org/wp-content/uploads/digital-library/09287.41281.pdf%7D`.

[72] Klaus Oberauer and Stephan Lewandowsky. "Forgetting in immediate serial recall: Decay, temporal distinctiveness, or interference?" In: *Psychological Review* 115.3 (2008), pp. 544–576. DOI: `10.1037/0033-295x.115.3.544`.

[73] Sergio Ocio and José Brugos. "Multi-agent Systems and Sandbox Games". In: *Multi-agent Systems and Sandbox Games*. 2009.

[74] M. Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems*. Vol. 2. Springer, 1999.

[76] Judea Pearl. "Heuristics: intelligent search strategies for computer problem solving". In: (1984).

[77] PriceWaterhouseCooper. *Global Entertainment & Media Outlook 2021-2025*. `https://www.pwc.com/gx/en/industries/tmt/media/outlook/segment-findings.html`. Accessed: 2021-09-21.

[78] J. R. Rayfield. "What Is a Story?" In: *American Anthropologist* 74.5 (Oct. 1972), pp. 1085–1106. DOI: `10.1525/aa.1972.74.5.02a00040`.

[80] Robbert van Renesse and Fred B. Schneider. "Chain Replication for Supporting High Throughput and Availability." In: *OSDI*. Vol. 4. 91–104. 2004.

[81] Ann Rigney. "The Point of Stories: On Narrative Communication and Its Cognitive Functions". In: *Poetics Today* 13.2 (1992), p. 263. DOI: 10.2307/1772533.

[82] B.D. Ruben. *Information and Behavior*. Information and Behavior. Transaction Publishers, 1985. ISBN: 9781412826259. URL: https://books.google.de/books?id=yVGOBI1qZRcC.

[83] Marie-Laure Ryan, James Ruppert, and John W. Bernet. *Narrative across media: The languages of storytelling*. U of Nebraska Press, 2004.

[84] Harold Scheub. *Story*. Univ of Wisconsin Press, 1998.

[85] Secret Lab. *Yarn Spinner*. https://yarnspinner.dev/. Accessed: 2021-09-29.

[86] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[87] Claude Elwood Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[88] Lauralee Sherwood. *Human physiology: from cells to systems*. Cengage learning, 2015.

[89] Sid Shuman. *Hermen Hulst Q&A: Interview with Head of PlayStation's Worldwide Studios*. https://blog.playstation.com/2020/03/10/hermen-hulst-qa-interview-with-head-of-playstations-worldwide-studios/. Accessed: 2021-09-13.

[90] Miguel Sicart. "Defining game mechanics". In: *Game Studies* 8.2 (2008), pp. 1–14.

[91] Divya Siddarth et al. "Who Watches the Watchmen? A Review of Subjective Approaches for Sybil-Resistance in Proof of Personhood Protocols". In: *Frontiers in Blockchain* 3 (2020), p. 46. ISSN: 2624-7852. DOI: 10.3389/fbloc.2020.590171. URL: https://www.frontiersin.org/article/10.3389/fbloc.2020.590171.

[92] Sony Interactive Entertainment LLC. *PlayStation Studios*. https://www.playstation.com/en-us/corporate/playstation-studios/. Accessed: 2021-09-13.

[93] A. Spearing. "Public and Private Spaces in Sir Gawain and the Green Knight". In: *Arthuriana* 4 (1994), pp. 138–145.

[94] Maarten van Steen and A. Tanenbaum. "Distributed systems principles and paradigms". In: *Network* 2 (2002), p. 28.

[95] G. J. Stephens, L. J. Silbert, and U. Hasson. "Speaker-listener neural coupling underlies successful communication". In: *Proceedings of the National Academy of Sciences* 107.32 (July 2010), pp. 14425–14430. DOI: 10.1073/pnas.1008662107.

[96] Unity Technologies. *Unity*. https://unity.com/. Accessed: 2021-09-29.

[97] Matt Weisfeld. *The object-oriented thought process*. Pearson Education, 2008.

[98] Wayne Weiten. *Psychology: Themes and variations*. Cengage Learning, 2021.

[99] WePC.com. *Video Game Industry Statistics, Trends and Data In 2021*. https://www.wepc.com/news/video-game-statistics/. Accessed: 2021-09-13.

[100] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[101] Michael Wooldridge and Nicholas R. Jennings. "Intelligent agents: theory and practice". In: *The Knowledge Engineering Review* 10.2 (June 1995), pp. 115–152. DOI: 10.1017/s0269888900008122.

[102] Robert S. Wyer, Jr. *Knowledge and Memory: The Real Story: Advances in Social Cognition, Volume VIII*. Psychology Press, 2014.

Ludography

[7] Bethesda Game Studios. *The Elder Scrolls V: Skyrim*. Microsoft Windows, PlayStation 4, Xbox One, MacOS, Linux, Stadia, PlayStation 5, Xbox Series X/S. 2011.

[23] DigixArt. *Road 96*. Microsoft Windows, Nintendo Switch. 2021.

[29] Eidos Montréal. *Shadow of the Tomb Raider*. Microsoft Windows, PlayStation 4, Xbox One, MacOS, Linux, Stadia, PlayStation 5, Xbox Series X/S. 2019.

[48] Jakob Raith. *Prototype Game*. Microsoft Windows. 2021.

[57] Larian Studios. *Divinity: Original Sin II*. Microsoft Windows, macOS, iPad, PlayStation 4, Xbox One, Nintendo Switch. 2017.

[61] Ludomotion. *Unexplored 2: The Wayfarer's Legacy*. Microsoft Windows, Xbox One, Xbox Series X/S. 2021.

[66] Mossmouth LLC. *Spelunky*. Microsoft Windows, Xbox 360, PlayStation 3, PlayStation 4, PlayStation Vita, Chrome OS, Nintendo Switch. 2012.

[69] Nintendo EAD. *Super Mario 64*. Nintendo 64. 1996.

[70] Nintendo EPD. *The Legend of Zelda: Breath of the Wild*. Wii U, Nintendo Switch. 2017.

[75] Paradox Development Studio. *Crusader Kings III*. Microsoft Windows, macOS, Linux, PlayStation 5, Xbox Series X/S. 2020.

[79] Red Hook Studios. *Darkest Dungeon*. Microsoft Windows, macOS, Linux, iOS, PlayStation 4, PlayStation Vita, Xbox One, Nintendo Switch. 2016.