CHALMERS | UNIVERSITY OF GOTHENBURG

Deadly Banquet

Creating believable non-player characters

Bachelor of Science Thesis in Computer Science and Engineering

JOAKIM SUNDLING
GUSTAV AXELSSON
HAMPUS NILSSON
TOM DAHLÉN
KARIN WIBERGH
MATHIAS CARLSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

Bachelor of Science Thesis

Deadly Banquet

Creating believable non-player characters

JOAKIM SUNDLING
GUSTAV AXELSSON
HAMPUS NILSSON
TOM DAHLÉN
KARIN WIBERGH
MATHIAS CARLSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
University of Gothenburg

Göteborg, Sweden 2016

Deadly Banquet
Creating believable non-player characters
JOAKIM SUNDLING
GUSTAV AXELSSON
HAMPUS NILSSON
TOM DAHLÉN
KARIN WIBERGH
MATHIAS CARLSSON

Examiner: NIKLAS BROBERG

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg 2016

Deadly Banquet
Creating believable non-player characters

JOAKIM SUNDLING
GUSTAV AXELSSON
HAMPUS NILSSON
TOM DAHLÉN
KARIN WIBERGH
MATHIAS CARLSSON
Department of Computer Science and Engineering,
Chalmers University of Technology
University of Gothenburg

Bachelor of Science Thesis

ABSTRACT

NPCs (non-player characters) in computer games are often very predictable and obviously artificial, which might be the result of developers devoting more resources to maintaining high-end graphics instead of creating believable NPCs. With the graphics available today already being very realistic, upgrading the AI (artificial intelligence) controlling the NPCs could be one way of creating a game that differs from the rest. This report documents the development of a game called Deadly Banquet, a murder mystery which focuses on the interaction between the player and the NPCs. All necessary building blocks for the AI which controls the NPCs are described, as well as the components for making the game framework. Further, diverse definitions of believability are discussed and an overview of methods for evaluating them is presented. Conclusions drawn from the project indicate the need for extensive memories for each NPC. They also indicate that using relations between NPC:s in decision-making could enhance their believability.

SAMMANFATTNING

NPCer (icke spelbara figurerer) i datorspel är ofta förutsägbara och tydligt artificiella, vilket kan vara ett resultat av att utvecklare spenderar mer tid och resurser på att upprätthålla en hög grafisk standard istället för att skapa trovärdiga NPCer. Eftersom standarden på grafik idag har blivit mycket realistisk så kan utveckling inom den AI (artificiell intelligens) som kontrollerar NPCerna vara ett sätt att särskilja ett spel från andra. Denna rapport dokumenterar utvecklingen av ett spel med namnet Deadly Banquet. Detta spel går ut på att lösa en mordgåta och fokuserar på konversationer mellan spelaren och NPCerna. Alla delar av AIn som kontrollerar NPCerna kommer beskrivas liksom de komponenter som bygger upp spelet. Diverse definitioner av trovärdighet kommer diskuteras och övergripande metoder för att värdera dessa kommer att presenteras. Projektets slutsats pekar på att NPCer behöver egna minnen för att förbättra sin trovärdighet. Även förmågan att göra val baserat på relationer mellan NPCer kan förbättra deras trovärdighet.

ACKNOWLEDGEMENTS

In this project some creative commons copyrighted material was used. Below follows the list of materials we used and the original author.

Tiles and sprites - Lunarea and Ayene
The dead body tile - Sanggameboy and Enterbrain

We would also like to say a special thank you to Thommy Eriksson, our supervisor, who helped us discuss when we strayed from our path.

Table of Contents

Clarification of terms

User = the human being playing the game

Player = the figure on the screen which is controlled by the user.

Character = A character in the game, includes both the player and NPCs

AI = intelligence exhibited by machines.

NPC = Non-player-character, a character not controlled by a user, but instead by AI.

1 Introduction

Non-player characters (NPCs) in games are generally extremely predictable and exhibit very shallow behaviour. While this is not a problem in games like Pac-Man, it can become rather annoying in games with a strong social component such as “Mass Effect”. This report will attempt to explore the difficulties causing this problem, as well as detail partial solutions.

1.1 Background

Interactive agents are computer programs which attempt to converse with their users. Most such programs are used as virtual receptionists or similar who help users find information, or are attempts to pass the Turing test(1). This test involves fooling a panel of human judges into mistaking a computer for a human during a short conversation, and requires the machine to be able to give human-like answers to a large and diverse group of subjects.

Common to all these aims is that much focus is on parsing user input and creating informative or convincing responses. Noticeably absent are things such as personality, initiative and social relationships, qualities which are particularly important for developing a believable agent, defined as "one who seems lifelike, whose actions make sense, who allows you to suspend disbelief. This is not the same thing as realism. For example, Bugs Bunny is a believable character, but not a realistic character"(2). Believable agents, defined in this particular way, could be used to substantially improve computer games which have a strong social component.

Role playing and adventure games often host a multitude of so-called Non-Player Characters (NPCs), which are game characters controlled by the computer rather than by a human. However, as a large proportion of the resources available to high-budget games - whether in terms of programmer or CPU time - are routinely assigned to maintaining cutting-edge graphics(3), they are often rigid and predictable. However, since graphics have reached a high level of realism in the top budget games, and further improvements will be less and less noticeable, game developers will most likely be forced to find other means of differentiating their games from others. Upgrading the AI controlling the NPCs is one very plausible direction of improvement, since not only does more advanced AI exist in certain games, but AI research in robotics and mechanics can also be used to further their development.

1.2 Purpose and objectives

This project aims to explore the components necessary to produce believable NPCs and to create an interactable game that focuses on dialogues to test these components.

More specifically, it attempts to answer the question: What makes an NPC able to have a believable dialogue with the player? To answer this, the following questions need to be answered: How does one measure believability? Which factors, such as character, current mood or situation need to be modelled, and how should they be modelled? How does one turn the intended meaning faithfully into words?

The project also attempts to explore how to create a functional game without prior experience within that particular field. How does one structure the game for easy extension and debugging? How should it be designed? How will the cooperation between the solutions to the questions posed in the previous section and the game be implemented?

1.3 Game Summary

The game is a murder mystery, where the player's goal is to figure out who the murderer is. The player is trapped with some other persons in a mansion. To figure out who the murderer is the player can talk to the other persons and cross-check their stories. When the user thinks he or she knows who the murderer is, the user will get one chance to select a suspect and end the game.

1.4 What is believability?

The use of the concept believability was chosen over realism because realism would require a great deal of insight into how actual human beings think, which is outside the scope of a pure computer science project. Also, there might be a conflict between realism and believability in cases where many people have a mistaken idea of what constitutes normal skills and behaviour. An example of this could be the popular misconception that eyewitnesses who are very sure of their identification of a suspect are actually correct(4). In this case, an NPC who sounds quite certain of something which the user later finds out is wrong might be seen as unbelievable even though the behaviour is quite realistic. What is wanted in a game for the general public is normally believability, not realism.

As stated in the introduction, the definition of believable agents is: "one who seems lifelike, whose actions make sense, who allows you to suspend disbelief." A case has been made for dividing the question of believability into player believability and character believability(5)(6). Player believability is the illusion that a human being is controlling a character, while character believability is the illusion of life - the idea that the character is a real, live, thinking, feeling robot or NPC instead of a human. An example of character believability would be a robot receptionist which says that it once dated the Pittsburgh stadium's scoreboard(7).

The difference between player believability and character believability could be described by looking at the NPCs in The Sims and Counter-Strike. In The Sims the characters are made to be believable as Sims, but do not try to simulate another human player. In Counter-Strike, the NPCs instead try to simulate another human player. The Sims' NPCs are character believable, and the Counter-Strike NPCs are player believable.

Player believability is maybe the most intuitive approach; at any rate, the user test proposed early in the project before any serious reading about believability was done tested for player believability; it included a trick question which asked subjects to guess which NPC was being controlled by the assistant. For a more detailed description of this test, see Appendix A.

However, character believability is a more suitable definition for testing this particular game against, because the capacities of the player and the NPCs vary (only an NPC can murder people, for example). For player believability to hold, the user would have to believe that there is a different way of starting the game such that a human takes on the role currently played by an NPC, and also that this human or humans then take it into their heads to play out a whole murder drama for the entertainment of the user. Therefore, in this project, we will be focusing on character believability.

1.5 Scope and Limitations

The purpose described above is a very wide question, and some specification as to the scope of our particular project is required. As mentioned in the background, game programming has a lot of time consuming parts and there is therefore a necessity for specific limitations.

1.5.1 Game limitations

The area of the game was confined to rooms within a mansion with no way to go outside, a constraint introduced to steer the workload away from the graphics and map-design. Each room represents a single game screen, and the characters are able to move freely between them.

1.5.2 What should be believable?

It was decided that this project would focus on the believability of NPCs during conversation, rather than believability achieved through detailed visuals and human-like movement. Visuals would occupy too much time to get up to any resemblance of believability and showcasing certain parts of human-like behaviour without good visuals would be very challenging. Visual cues such as body language and facial expressions, for instance, are not possible with such a type of light-weight visuals and were therefore excluded from the scope of this project.

1.5.3 Conversations

As the scope is geared towards conversation there is also a need to address the shape of these conversations. Although the intuitive solution would be to let the user write what they want to say and then get responses, this option was discarded in favour of a type of sentence-building through selection of parts. Again this was mainly due to time constraints as only creating a parser to interpret free text would be very demanding, and would have steered the problem away from the main purpose. The chosen type of conversation system also allows for a more controlled environment which can eliminate the possibilities of edge-case scenarios which could break the believability of the NPC.

The psychology related to conversations and relations is a part of the project of minor importance. It was decided that the necessary information on these types of nonengineering disciplines would be acquired from external sources rather than created through experiments/surveys. The reason for this is the lack of a member with that type of prerequisite knowledge.

2 Method

This section provides an overview of the tools and materials used in this project. It also sheds some light on some concepts used in for instance program structuring, and gives a slight motivation for why these tools, concepts and materials were chosen.

2.1 External tools

It was decided to base the game on the java-library Slick2D[1]. This decision was made mainly because Slick2D is fast to set up and it is easy to get a working version running. It was also due to the library being flexible enough for the purpose, and containing some pre-made utilities such as pathfinding to ease the creation process. To make the graphic design as efficient as possible an auxiliary program called Tiled[2] was used in the process of map-creation. This was partly due to the ease with which this program can make and modify maps, and partly due to Slick2D having built-in support to use the creations.

2.2 Version control - Coding in a group environment

With coding being the paramount task, a type of version-control was needed in order to support multiple people coding on the same project, possibly at the same time. The service that was decided upon, GitHub[3], is widely used and all members of the project had used it prior to the start of this project. Using this service solved the problem associated with needing to manually send and merge files, and also provided a backup with support for rolling back the project code to earlier versions.

2.3 Information gathering

As mentioned in the limitations section, some of the questions posed need some input from other disciplines such as psychology. It is as stated not within the scope of this project to do any of our own research, so all the information pertaining to these areas was acquired through literature.

Literature, forums, and other sources of information were also used in the process of structuring, planning and coding, so as to find helpful examples of solutions to issues similar to those encountered here. This type of information ranged from the simple forum references regarding small programming issues to research-level text about text-based AI.

[1] For more information about Slick2D: http://www.slick.ninjacave.com/
[2] For more information about Tiled: http://www.mapeditor.org/
[3] For more information about GitHub: https://github.com/

2.4 Use Cases

Use Cases were created to define what it would be possible to do in the system. They cover for example how the characters move, how characters talk to each other and how the conversations between characters should work. The creation of use cases was useful for the group to agree on specific details that were still unclear. All the Use Cases can be found in appendix B.

2.5 Program structure and creation

The program itself was made through an iterative process. This means that the program was created one version at a time, and gradually improved until its current state. Although this makes the program structure subject to change in each iteration, some kind of general structure was still required. This major structure was decided to be based upon the well-known Model-View-Controller model. As this model mostly consists of guidelines on how to separate the program structure, rather than something concrete, the specification of our implementation of it will be described later in the report. The reason such a structure was used was partly due to the familiarity with it present in the group, and partly due to the ease of debugging and scaling it gives.

2.6 Choice of AI type

Some different methods of AI implementation were investigated and assessed on their compatibility with the purpose of this project. In the game an NPC will need to be able to move, converse, and idle. This leads to a need for at least three different behavioural patterns, also known as states.

Behavioural tree was a method which was excluded almost instantly. The method is an extension of the state-based AI that is presented below, and was made to combat the problems that arise out of having a high number of states. Since only three states were needed in this specific instance, this method was considered unnecessarily complicated.

Neural network was also considered too complicated and somewhat unsuited to this type of problem. Neural networks are non-deterministic and also require a concrete goal to strive for. With an insubstantial term such as “believability” as the purpose of the project, such an uncontrolled structure was quickly excluded from the deliberations.

The last type of AI structure investigated was a state-based AI and it was also the one that was later used. Since the focus of this project was about the dialogue, and everything therein would happen in a single state, there was no need for more complicated methods. Only three states, based upon the actions mentioned previously, were necessary, and as such none of the downsides of the state-based AI was of any importance to this project. With this in mind and considering that it is one of the simplest one to get into and implement, this became the method of choice.

3 The project work - planning and coding the game

As previously mentioned the version control service used is Github, where the game can be found[4]. The coding was done iteratively as stated above, with weekly meetings, where problems were brought up, and solutions to last week's problem presented. Before the implementation started some brief research was conducted. The game design was decided partly based on the use cases found in Appendix B.

As of now the conversations do not give the impression that a murder has just occurred. However, the basis of the questions is derived from this knowledge. For example the questions the player is able to ask the NPCs are about the location of other people, about who did something and about what someone's opinion of someone is. The plan was that all of these question would help the user to determine who the murderer is.

The coding process was divided into different areas, each concerning different parts of the game. The parts were then worked on separately by the members of the group in order to get a deeper understanding of each part. As the project progressed the different parts were combined and started communicating with each other. By the end of the project all parts were brought together in order to make the game functional.

3.1 Game framework

This project was largely focused on the creation and implementation of the game and AI, and a solid game framework with a controlled structure was considered an important cornerstone. Through in-house planning and use of a known method of program structuring, Model-View-Controller (MVC), a program structure for the game was decided upon. This structure heavily emphasized a controlled environment which was achieved by forcing all actions that would affect the characters and game world to be executed through the World instead of for example a character moving themselves. In this way the World, which has access to practically all the data, could use this to check for collisions, create paths and overall enforce that no unintended actions were allowed.

Visuals and the control of the actions characters would try to perform were clearly separated into their own respective parts. This adheres to the way an MVC structure is commonly interpreted and also gave an even more controlled environment since all game-altering changes needed to be invoked through one of the three parts mentioned above.

One of the reasons such a design was decided upon even though the coding process becomes slightly more arduous, was the need for easier debugging because of all the small and constantly evolving parts. Another was that it simplifies the code division within a group. The following subsections will present a more in-depth description of the individual parts that the framework ended up containing and the different design decisions made within each one of these.

[4] The project repo can be found at : https://github.com/Mathcar/DeadlyBanquet.git

3.1.1 State based Game

The overall design of the game was implemented in a way that makes the game, at any point in time, be in one of several states. This design also makes it possible to transition between these states when certain conditions are met. The game can in other words be treated as a finite-state machine[5]. This isolation of states allows for each state to be easily managed and maintained. Slick2D, the utilized java-library, also has extensive support for state-based games which further motivated the decision to design the game this way.

For this project a total of four states were created; Menu, Game, Conversation and Endgame. They are shown in figure 1.

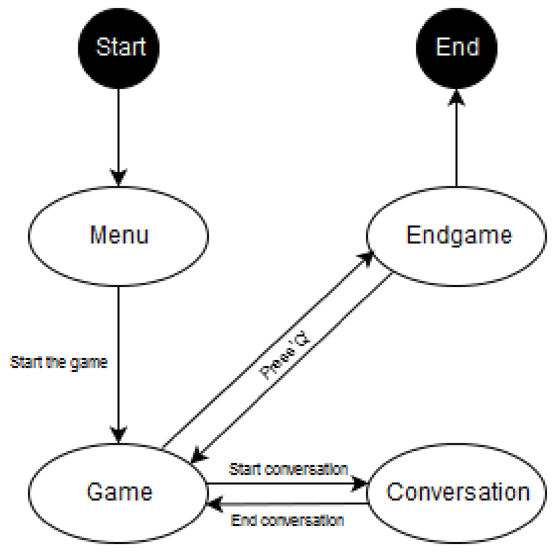


Figure 1: The state transitions and the triggers for each transition (The black dots represent opening and closing the game)

3.1.2 MVC

To allow a user to use the program a user interface was needed and the design pattern Model-View-Controller(MVC) was used to implement that interface. This pattern means that the underlying model is separated from what is shown to the user and a controller is the only way to modify the model. The view gets the information it needs from the model to present a graphical representation of the model to the user.

This project is using a variant that strongly connects the model and controller. The classes that represent the model also contain methods to modify it. The main flow of the program is to first create the world, then alternately update and render it from the main game class.

[5] Description can be found at: http://www.cse.chalmers.se/~coquand/AUTOMATA/book.pdf

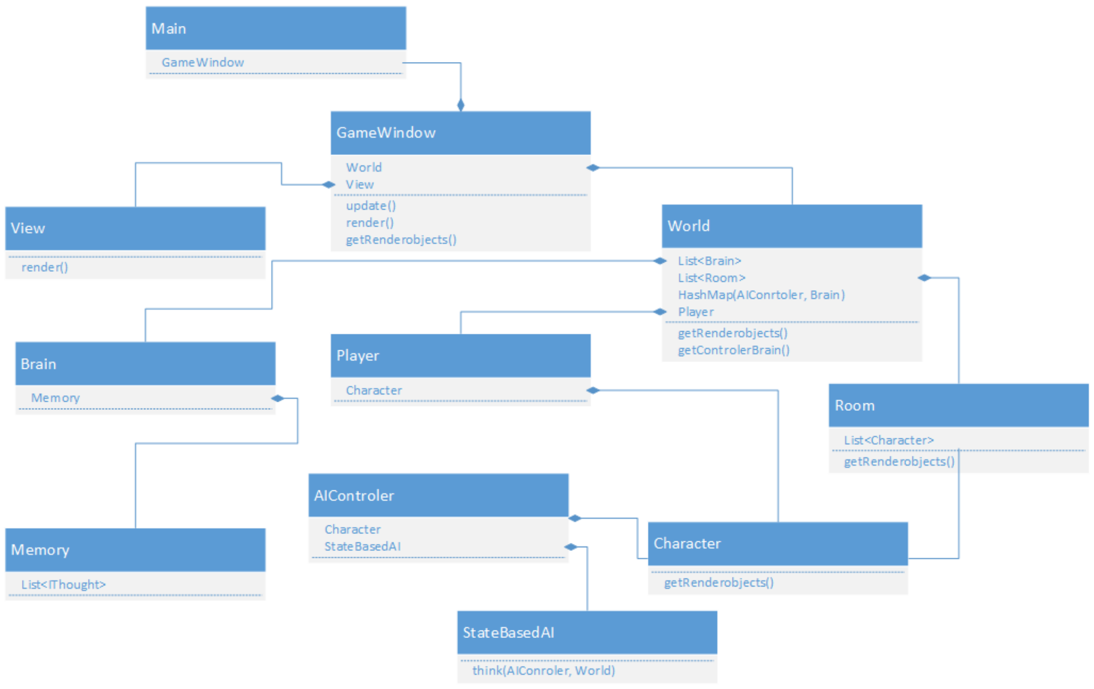


Figure 2: An abbreviation of the class diagram of the project structure

3.1.3 Physical Model

The “physical model”, the entity containing the data of all the things in the program, was implemented in a hierarchical manner. The top level is known as the World and contains the controller interfaces for both the AI and the player, namely AIController and Player. It also owns all the Rooms, which in turn have a shared ownership over the characters currently in them. For a more detailed overview of the set-up see Figure 2 above.

Much of the control of the program is localized in the top level of this model, since all actions need to be petitioned to the World and are thereafter executed by it if they are considered valid. The model also contains classes such as Character which is the simple collection of data concerning the physical attributes of a person, and Room which contains the data regarding a single room. These only act as auxiliaries and all their actions are always executed through the World.

3.1.4 State Based AI

The AI in this project is responsible for deciding what the NPCs will do as well as for delegating the execution of these actions to the World. The core of the AI is a State-Based AI (SBAI) which is responsible for the decision making and scheduling. This is a common method when creating AI for games due to its simplicity and efficiency(8). When constructing the SBAI for this project a lot of inspiration came from the Valves community developer wiki(9).

An SBAI uses finite automata to determine what the AI should do. The SBAI used in this project has three states: Idle, Moving and Talking. During every update the AI generates a list of everything it can see and a list of conditions. The conditions are generated from a combination of what the AI can see and what it remembers. The conditions are then used to determine the transitions between the states in the automaton of the AI. If the SBAI does not have a schedule, or is interrupted, a new schedule is generated depending on the current state and the conditions by combining a number of tasks to achieve a goal. It then executes the first task in the schedule.

All the decisions are made by the AI by generating a random number and checking if it is larger than a certain threshold, similar to rolling a dice. Most of the decisions made by the AI is done within the Idle state mostly because the AI reverts to the idle state when it finishes a schedule or gets interrupted. Within the Idle state, for every other character in the room, it generates a schedule to walk up to and talk to that character if it passes a threshold test. If none of the tests succeed it does the same test for every door in the room and generates a schedule for walking through that door. If it still has no schedule it checks if it has a condition that it is standing close to a door, and schedules moving away from that door if it is.

These threshold tests were not the intended end-product; rather the decisions were intended to be based upon relations, drives and short-sighted goals. But as this goal was slightly out of reach and the SBAI still needed to do things, a temporary threshold-test solution was implemented.

Although this system is a very common one, there are some slight differences to account for. The SBAI used here was extremely simple in terms of the number of states it could take on, and all states except for the one taking care of conversations were very small. The focus in this project was not on the structure of the AI, but rather on that single state which would control the responses of the NPC, since that was what could afford the NPC:s believability within conversations.

3.1.5 Control Interfaces

The SBAI is paired with the controller interface called AIController. The role of the AIController is to act as intermediary between the World and the decision-making SBAI. The reason such an intermediary was introduced was mainly the necessity of limiting the accessible information for the SBAI to simulate the believability the project aimed for.

This AIController works by sending requests to the World to perform the actions the SBAI has decided upon and then reporting the result of the attempt back to the SBAI. It also receives all useable information for the AI from the World. The amount of information intended to be recorded and/or used by the AI can as such be regulated by simply modifying this particular interface.

The information that passes through is mainly of the observable kind. The AI receives information regarding the whereabouts of other NPCs, and any interactions, within its current room. The conversation also passes through here, but is not checked or altered, being parsed later on instead.

The player has a similar setup, but the intermediary is instead between the player input and the model. This Player intermediary also passes information to the World in a similar manner to the one mentioned above.

3.1.6 Pathfinding within a room

For the NPC:s to be believable, they need to be able to undertake their own actions. This necessitates pathfinding as a part of the model, as walking through bookshelves and/or teleporting lends the NPC:s a lot less believability. The pathfinding uses an algorithm commonly known as A*(A-Star), which directs the path-searching by using a heuristic so as to avoid having to probe the entire grid every time(10). This is a combination of the best parts of the greedy best-first search and Djikstra's algorithm, which are both well-known solutions to pathfinding problems.

The specific implementation that was used in this project can be found in Slick2d. As the areas where movement is possible are fairly uncomplicated, the most basic heuristic of just the total distance to the target point was used to find the best path.

3.1.7 Intra-Room pathfinding (MasterPath)

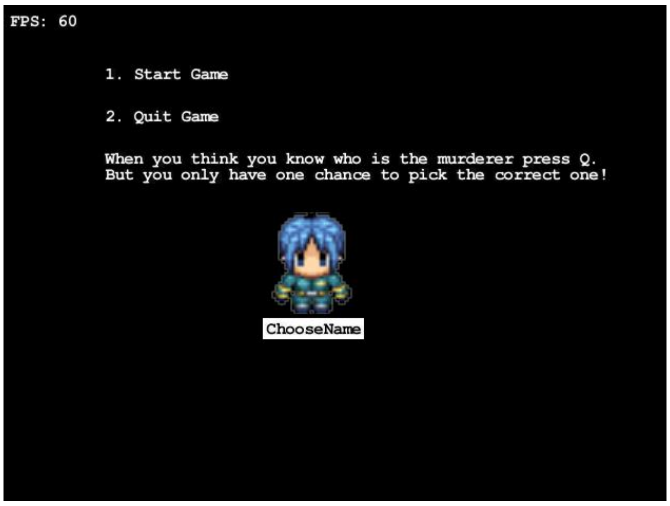
Each room is treated as a separate entity with its own grid and the normal pathfinding within a room does not encompass finding paths between rooms since it only works on a separate grid. Therefore a top level MasterPath was used in parallel with the room-specific pathfinding to allow characters to leave and enter rooms. The MasterPath can be seen as a list consisting of all the rooms that need to be crossed in order to reach the destination. This MasterPath could then be used together with the pathfinding within a room to locate the correct door to go through. The MasterPath was also derived using A* but on the layout of the mansion instead of the grid of a separate room.

3.1.8 Visuals

As mentioned above, the main focus of the project was the believability of the AI and the decision was made to make the graphics a lower priority. As stated in the method, the program Tiled was used in the creation of the maps. This program creates a layer-based tmx file, which is the native file format used to describe tile based maps. The map is based on so-called tiles, 32x32 pixel squares which the character moves on. One advantage of layer-based programs is that they make it easy to decide if a character can walk on a tile or if it is supposed to be blocked. Also, if all the doors of the rooms are placed in one layer they are easy to access when creating the room changing methods. Work on the graphics started small with the creation of one single room, the living room, but then extended to the creation of two more rooms: the kitchen and the bedroom. The ambition to make custom-made pixel art was given up early in the project as it appeared to be very time-consuming. Instead the project made use of art that falls within the lines of Creative Commons.

Work on the graphics was done alongside with the coding. As the development of the game proceeded, more graphical components were added. As mentioned above the project started with a single room, the living room, and a static smiley face as the player. Later on a more realistic image was added as the first NPC. At that time the player was unable to interact with the NPC and it had basically no function. As the project progressed the static smiley was replaced with a sprite that has animations for walking in all four directions. The need for more rooms came and both the bedroom and the kitchen were created. As the code for room changing was written doors were created and the layout for the mansion was established. The NPC became a sprite as well with all the animations and later 5 other unique NPCs were added into the game and placed in the different rooms. The last thing that was done in the project graphics-wise was that a dead body was added in the bedroom.

As mentioned earlier the game was designed as a state-based game and each of these states have their own visuals.The Menu is where one picks the name of one's character and starts the game. The user is greeted with a picture of his or her character and the options to quit or start the game as seen Figure 3.



```
FPS: 60

1. Start Game

2. Quit Game

When you think you know who is the murderer press Q.
But you only have one chance to pick the correct one!
```

Figure 3: The menu (starting screen) of the game.

The next visual state is the Game state. Here, the user can control their character and interact with the NPCs in the mansion. This character can move between the rooms shown in figures 4, 5 and 6 in order to figure out who the murderer is.

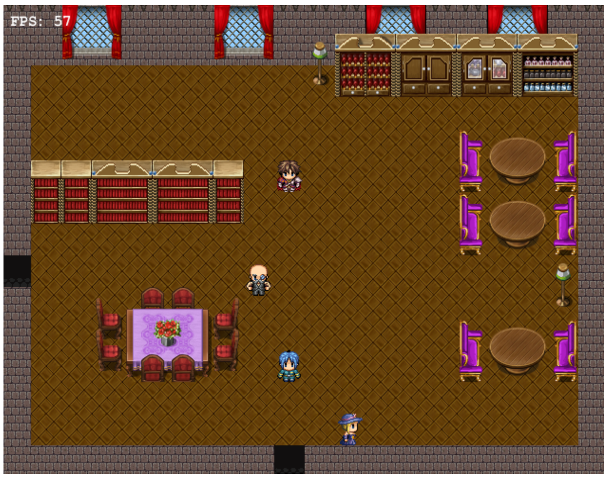


Figure 4: The living room

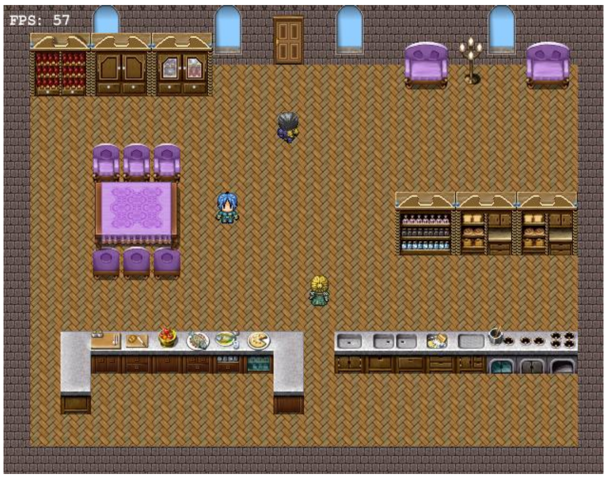


Figure 5: The kitchen

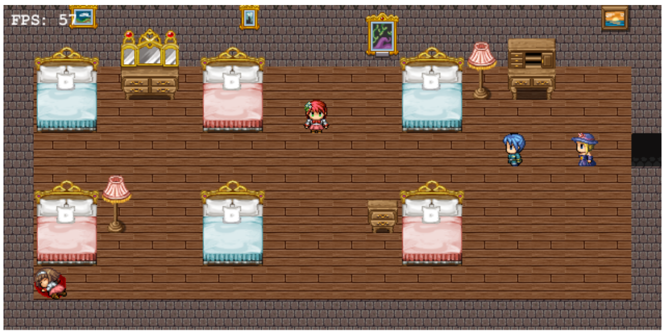
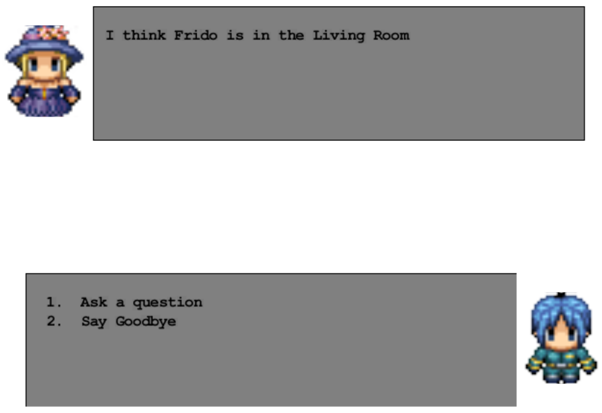


Figure 6: The bedroom

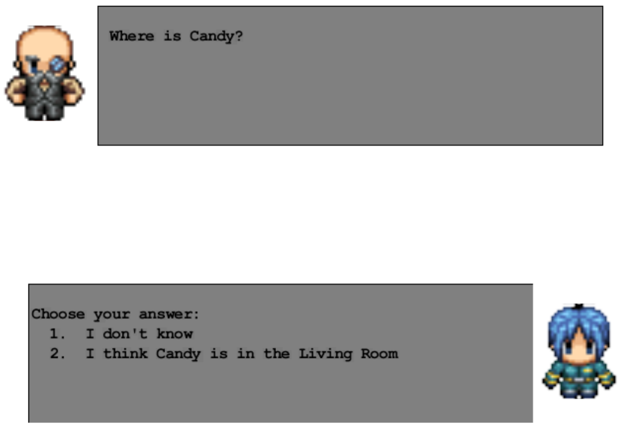
When the player talks to an NPC or when they start talking to the player, the Conversation state is entered, shown in figures 7 and 8. While in this state the user has different dialogue options, such as asking the NPC about someone's whereabouts, to choose between. The user can also ask the NPC about its opinion regarding any of the other NPCs. Lastly the user can ask the NPC about an event that has occurred. When the NPC engages in conversation with the player the same thing happens except that the user is the one to answer the questions.



I think Frido is in the Living Room

1. Ask a question
2. Say Goodbye

Figure 7: Conversation started by the player

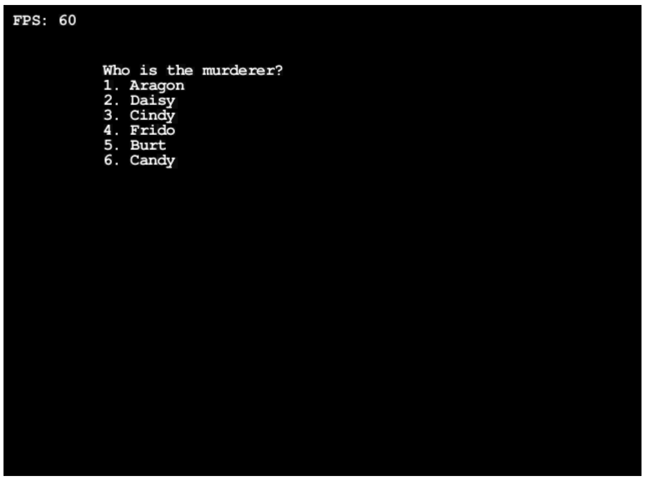


Where is Candy?

Choose your answer:
1. I don't know
2. I think Candy is in the Living Room

Figure 8: Conversation started by the NPC

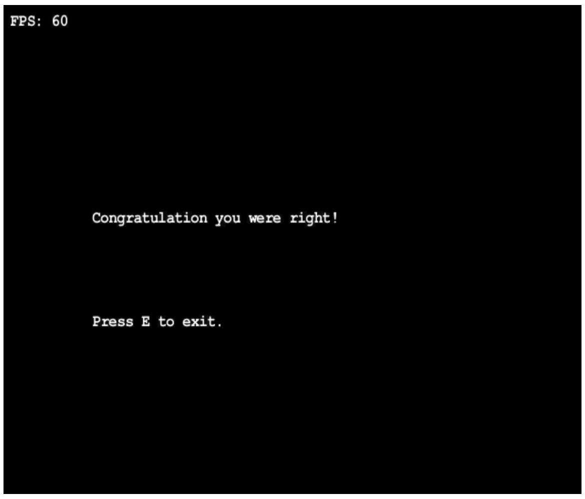
The last visual state is the Endgame state. Here the user either win or lose depending on whether they are able to accurately point out the murderer. As shown in figure 9 the user gets a list of all the NPC names and gets to pick who he or she thinks the murderer is.



```
FPS: 60

Who is the murderer?
1. Aragon
2. Daisy
3. Cindy
4. Frido
5. Burt
6. Candy
```

Figure 9: The screen where the user chooses who he or she thinks the murderer is

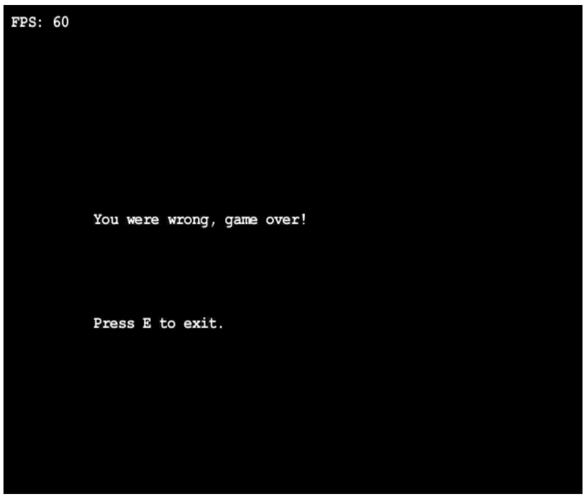


```
FPS: 60

Congratulation you were right!

Press E to exit.
```

Figure 10: The winning screen



```
FPS: 60

You were wrong, game over!

Press E to exit.
```

Figure 11: The losing screen

3.2 AI content generation and handling

The AI component which stores memories and controls the conversation of a particular NPC, here called the Brain, is produced in a factory. This factory will randomize traits such as character if none is supplied in order to improve the replay value of the game. There also exists a method which produces multiple Brains at once. This method, although not used in practice, is intended to let the other Brains-to-be know information, such as each character's whereabouts, about its twin Brains without that information having to be hard-coded each time.

It is self-evident that NPCs cannot think in human language the way we (presumably) do. Instead, they use a vastly simplified method of handling and sending around information. This method must be able to handle information storage and transformation as well as interpretation of incoming and assembling of outgoing information.

3.2.1 The types of information used by the AI

The NPCs handle a variety of information. Some of it is closely related to the research question and is here described in detail; information which is only relevant because of the particular subject of the game - such as who murdered whom - is omitted.

First and foremost, the NPCs need some representation of their current mood, their character (also known as temperament and defined as the average of moods over a long period of time), and their feelings about people and circumstances. For this, the PAD model by Albert Mehrabian (professor emeritus of psychology at UCLA) was used. This model consists of a cube, where each axis goes from -1 to 1. The three axes represent Pleasure, Arousal and Dominance. A mood, character or feeling is represented by a point in the cube(11). For more information about how particular values were interpreted, see Appendix C. Although the basic capacity for feeling and having opinions has thus been encoded, the game does not actually make much use of it due to time constraints.

Further, NPCs hold a set of beliefs about the world they inhabit. This set of beliefs must be distinct from the actual state of the world; for instance, it must be possible to hold the belief that Mary is in the kitchen even when Mary is somewhere else. In this project, these beliefs mainly concern people's whereabouts, actions, and interactions. This information can come from conversations or from an event which the NPCs observe directly; for example, this includes watching people move around the mansion and interacting with each other.

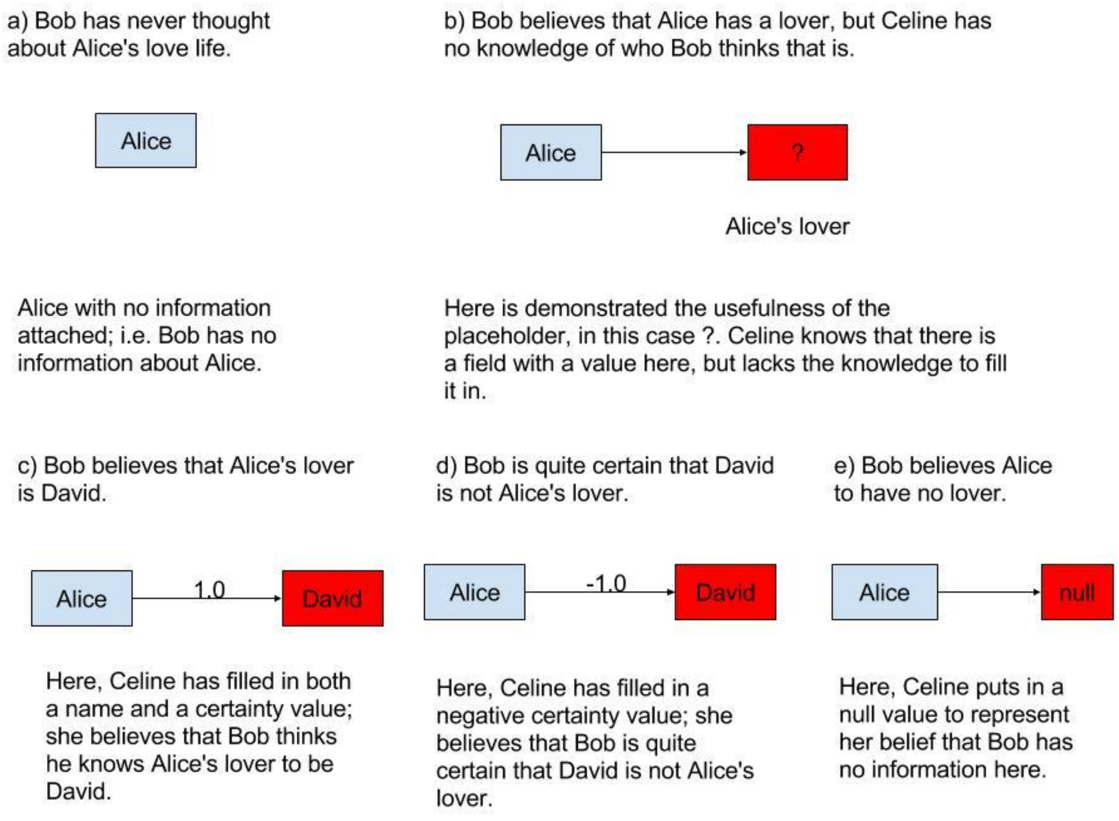
The last important feature for the NPCs is what is known as theory of mind(12), the ability to reflect about other people's thoughts and beliefs. It must be possible for NPC A to believe, for example, that NPC B believes something to be true even if A believes it to be false, or that B is in possession of information which A does not have.

3.2.2 Storing and retrieving information

We determined by a combination of reading and discussion that whatever format is used internally to represent information must have the following properties:

- It must be possible to pass on any piece of the information singly or in arbitrary combinations to another agent in the same way we humans can theoretically choose to impart as much or as little information of any kind as we please. The format's smallest units must therefore represent the smallest components of the information being represented. As this project does not include any general method for deconstructing arbitrary information into such units, the specific deconstruction used in it will not be described in this report; for details, see the implementation on GitHub.
- It must be possible to represent the idea that somebody has an idea about what value a variable has, without having to specify what value that person believes the variable to have. This is particularly useful if an NPC wants to ask a question - it is only common sense to ask somebody who one believes to be able to answer(13). Further, the opposites of these statements must be distinguished from the idea that one has never thought about the matter. An example of this is found in Figure 12.

- It must be possible to nest beliefs about others in an arbitrary number of levels ("Alice believes that Bob believes that Celine believes…")(13).
- Information pieces must be possible to combine in the same ways that humans do, for example if or when statements.



a) Bob has never thought about Alice's love life.

Alice with no information attached; i.e. Bob has no information about Alice.

b) Bob believes that Alice has a lover, but Celine has no knowledge of who Bob thinks that is.

Here is demonstrated the usefulness of the placeholder, in this case ?. Celine knows that there is a field with a value here, but lacks the knowledge to fill it in.

c) Bob believes that Alice's lover is David.

Here, Celine has filled in both a name and a certainty value; she believes that Bob thinks he knows Alice's lover to be David.

d) Bob is quite certain that David is not Alice's lover.

Here, Celine has filled in a negative certainty value; she believes that Bob is quite certain that David is not Alice's lover.

e) Bob believes Alice to have no lover.

Here, Celine puts in a null value to represent her belief that Bob has no information here.

Figure 12: Graphical representation of Celine's model of Bob's model of the world - i.e. what Celine believes is in Bob's head.

In this project, a piece of data which fulfils these requirements will be called an IThought; the name stems from the fact that in the example game, the structure is implemented as an interface. This name can refer both to a "smallest component" and to a combination of such components, such as an if statement.

In order to meet the second requirement, most IThoughts can be furnished with three things: a null value, a placeholder, and a certainty value. The null value will represent absence of information ("Bob believes that Celine does not know whether Alice has a lover”). This, of course, makes sense only at levels below the top one, since at the top level an absence of information is best modelled by an absence of the corresponding IThought. However, at lower levels it is needed to distinguish between "Bob is under the impression that Celine does not know whether Alice has a lover" and "Bob has not considered Celine's beliefs about whether Alice has a lover".
The placeholder represents the idea that the information exists in the specified place. It is used to distinguish the idea "Bob thinks that Celine knows who Alice's lover is, but does not know who Celine thinks the lover is" from the idea "Bob thinks that Celine

thinks that Alice's lover is David". In this implementation, the placeholder value is reused for questions, where it signifies the information which the speaker would like his hearer to fill in. This reduces the number of special values needed.

The certainty value, beyond distinguishing between "knows" (or rather, "believes themselves to know") and "thinks", can be used to turn a statement into its negation (for example by using a negative value): "Bob thinks that Celine knows who Alice's lover is" can then be turned into "Bob believes that Celine does not know who Alice's lover is." In the same manner, "Bob thinks that Celine thinks that Alice's lover is David" can thus be converted into "Bob thinks that Celine thinks that Alice's lover is not David, i.e. that either Alice's lover is somebody other than David or Alice has no lover."

The IThoughts are stored in an array, in which each piece of information is stored at most once. If the same piece of information has to be stored several times (such as Mary's whereabouts now and her whereabouts ten minutes ago), these items are made to form a sort of linked list sorted by time.

The elements of the array are not sorted in any particular manner; their order is in practice defined by the order in which they were added to the array. The original idea was to sort them by the absolute value of the certainty, so that the most salient information comes first in the list. However, due to a Java version problem on some computers the sorting code was commented out and it was subsequently discovered that even on machines where the code worked it did not make the faintest difference in practice. This circumstance, most likely caused by the small amount of information actually stored in the list, means that no sorting was ever actually used.

A find function is provided which given a fully or partially filled IThought returns all IThoughts in the memory which match that information. Partially filled in information is represented by a placeholder; this allows the NPC to search for answers to a direct question in its memory without deconstructing the question.

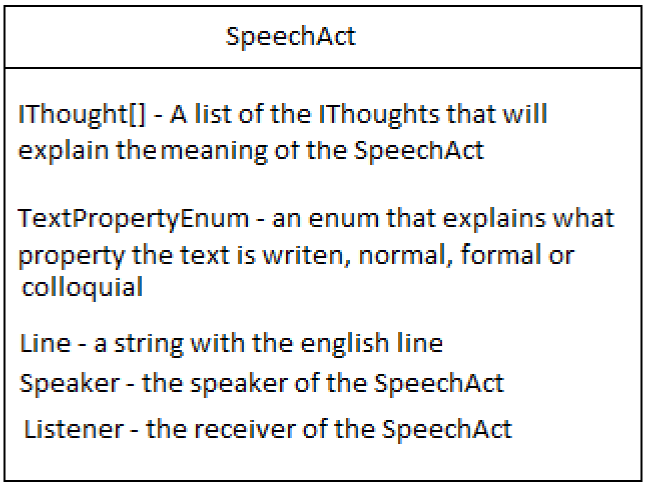
3.3 The dialogue system

Since the human dialogue system is very complex, an accurate representation of it is not an easy task, and could be a thesis on its own. The system that this project settled on is based on the assumption that there is always a leader of the conversation, who steers the conversation towards that person's desire. Another assumption was that the dialogues are based on questions and answers. In many video games this is the preferred way of doing it(14), since games often revolve around a player who is in constant need of information. However, in this project the NPCs could also be the leader of the conversation and ask the player questions, which he/she might answer or not.

3.3.1 SpeechActs

During any given act of communication between NPCs or between the player and an NPC, it is possible to give or receive several pieces of information at once. These are, together with the speaker and listener, encapsulated in a unit which will here be called SpeechActs.

Apart from the factual information - such as "We're snowed in" - a SpeechAct contains information about the utterance’s register; that is, whether it is proper, colloquial or neutral. Figure 13 shows what information a SpeechAct object contains. While it was originally intended that this information should be set by the NPCs, the less time-consuming option of hard-coding this value was used in the implementation.


SpeechAct

IThought[] - A list of the IThoughts that will explain the meaning of the SpeechAct

TextPropertyEnum - an enum that explains what property the text is writen, normal, formal or colloquial

Line - a string with the english line

Speaker - the speaker of the SpeechAct

Listener - the receiver of the SpeechAct

Figure 13: Model of what information a SpeechAct contains

Since all the SpeechActs, as in human dialogues, are basically transportation of a thought, the creation of a SpeechAct needs at least one IThought it can convert to readable English. This process takes place in the SpeechActFactory.

3.3.2 The SpeechActFactory

Given one or several IThoughts, information regarding the register of the text, the sender and the receiver, the SpeechActFactory will create a SpeechAct. There are multiple ways this could have been constructed. Since the user does not write in free text a parser was not needed. Since the user is offered alternatives to choose from, and the NPC wants to translate an IThought into English text, the simplest way seemed to be to create pre-made strings to match certain IThoughts.

To begin with, some text files were created. These text files were divided into question, info and greeting phrases. Each text file contains multiple rows, and each row contains the English string, and some atoms, simple variables which specify what type of statement it represents and how the statement is spoken. The decision to have these lines in text files and not in the Java files was made to make it as easy as possible to modify and add more lines.

Since it would not be reasonable to create exactly all of the lines that could ever be said, some of them contain token words which start with a #; these “hashtag-words” could be words like “#room”,”#item” and “#person”, and would later be replaced with an appropriate English word. For example, a line could look something like this: “#person is in the #room”, later the appropriate person and room would be filled in to create a complete statement, for instance “Cindy is in the kitchen”.

When an IThought needs to be converted to English, or more accurately a SpeechAct, it goes through the SpeechActFactory. Given an IThought, the SpeechActFactory would go through the appropriate list of the lines from the text files, and return a corresponding SpeechAct. This SpeechAct contains all necessary information for both the computer and the user to understand the meaning of it.

3.3.3 Receiving and sending information

The algorithm for NPC reply generation presented in figure 14 - which is a simplified pseudo-code version of the hear function in the game implementation - makes no claim to being in any way similar to that used by human beings. One notable difference, for example, is that it only responds to the last statement. This method was chosen because the alternative would be considerably trickier; not only because previous interactions must then be kept track of, but because more difficult questions then need to be answered, such as at which time in a conversation it becomes weird to refer to something said earlier because humans would not keep the information in their short-term memory more than a short while.

```
for each IThought i in incoming SpeechAct
        if i is a "don't know" message continue
        if i is a question
                attempt to find an answer in memory
                if that fails, generate a "don't know" message
              else
                calculate a sensible answer based on knowledge
                add the incoming information to memory
        add generated answer to list of possible answers
select from the list of possible answers the message most in line with the NPC's goals. If there is a
pressing goal which the suggested answers do not help reach, discard possible answers and
replace with goal-oriented message.
```

Figure 14: Pseudocode on how the NPC selects an answer to a SpeechAct.

Although this algorithm includes considering plans, the actual implementation never got as far as including a planning algorithm, so that this part of the algorithm was simplified to simply selecting all possible answers.

Further, this algorithm does not specify a meaning for "add incoming information to memory". For simplicity's sake, the current implementation simply adds the information "I know that the speaker believes that i is the case" to memory. This, of course, by no means represents all the inferences that can be drawn from a communication; other suggestions are found in the discussion in Section 5.

3.3.4 The menu system for constructing a SpeechAct

There are a lot of different ways of letting the user choose what to say; some of them give the user a lot of freedom in what to say. This can sound like a good thing, but the drawback is that it can be hard to steer any conversation in a certain direction if the user can always say whatever they want. An example of that type would be to let the user write in free text. A completely different way to do it would be to give the user a couple of choices which the game thinks are sensible responses. This option will do a good job in leading the conversation but will limit the user's free will.

This project wanted to make sure that the player felt free to say as much as possible. However, it was still important to be able to steer the dialogues in a way that matched the game atmosphere. The solution was to divide all the possible things a play should be able to say in categories and let the user select categories to come closer and closer to whatever the user wanted to say. For example, if the user wanted to ask about the location of a certain NPC, the user would first select the “ask a question” option, then the “ask about a person's whereabouts” and then select the character the question was asking about. When all this has been done the user will get a final choice of what to actually say. If the user wanted to ask about the whereabouts of Cindy the options would be:
“Where is Cindy?” - The neutral way to ask the question.
“Do you happen to know where Cindy is?” - The formal way to ask the question
“Where is Cindy hanging?” - The colloquial way to ask the question
In this last stage there is also always an option to abort, in case the user has mis-clicked or changed his or her mind.

In the case that the player is being asked a question the option to answer the question appears, if the player knows the answer, and an option to say “I don’t know”. This means that for now there is no way for the player to lie to an NPC; this could of course easily be incorporated in the game.

3.3.5 Dialogue overview

To give a clearer picture on the flow of the conversation system, see figure 15 below.

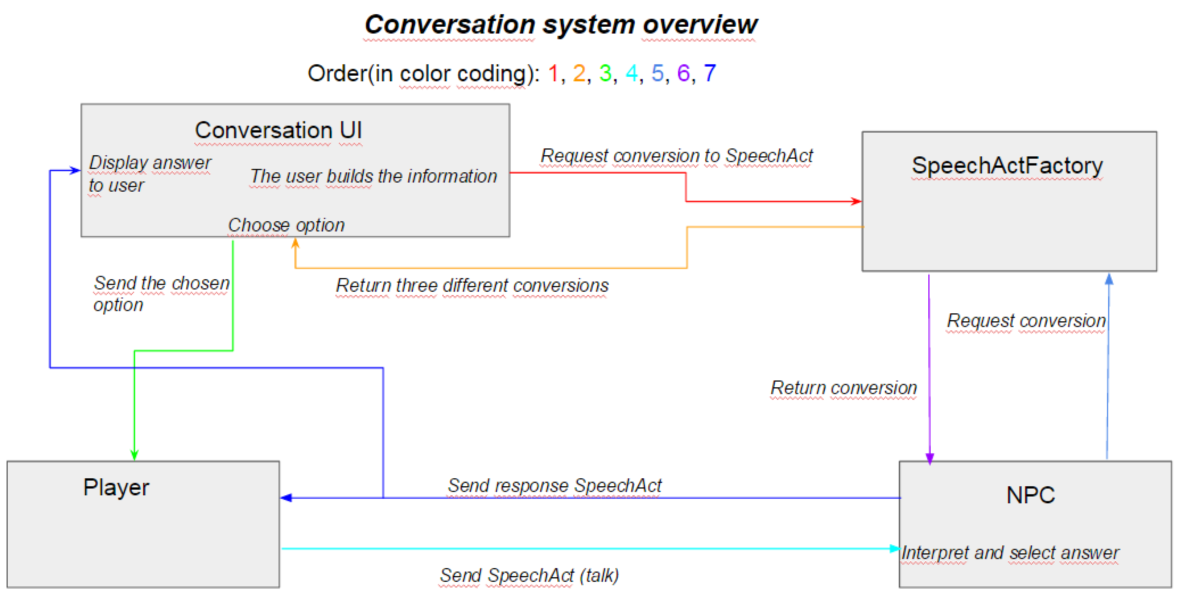

Conversation system overview
Order(in color coding): 1, 2, 3, 4, 5, 6, 7
Conversation UI
Display answer to user
The user builds the information
Choose option
Request conversion to SpeechAct
SpeechActFactory
Return three different conversions
Send the chosen option
Request conversion
Return conversion
Player
Send response SpeechAct
NPC
Interpret and select answer
Send SpeechAct (talk)

Figure 15: An overview of the conversation system

4 Results

The user can walk around in the game and talk to the NPCs. These NPCs can also move around in the game world using the previously mentioned pathfinding, and initiate conversations with either other NPCs or the player. The conversations between NPCs have a reliable structure but lack in variation and quantity of implemented content. The player conversing with an NPC flows well, but the amount of different IThoughts that can be handled by the system which allows the player to select what they want to say is currently rather limited. As for NPCs leading a conversation with the player, it is still a system in its infancy and as such can only handle statements and questions regarding the whereabouts of people.

The information the NPCs are currently aware of is the whereabouts of people they see enter or leave their current room, any interaction which happens in the current room and also whatever they learn from a conversation with someone else. Since it is as of now not possible for the player to pose a statement rather than a question to an NPC, all their information will be the result of questions they ask other characters.

The decisionmaking of the AI is semi-random mostly due to time constraints. It is a simple way to simulate some intelligence by having the NPCs move around and interact with one another. What was intended was that the SBAI should access the brain with its memory and consider that information when deciding on what action to take.

There are some extra pieces of the program which are not visible in the current game due to these not having been merged into the game structure. This merger is one of the objectives in the objectives section and is also one that was not completed in its entirety.

As both the implementation of the conversations and the memories of the NPCs are currently limited, the NPCs are not in any way perceived as believable through dialogues. Despite this, the process has given some valuable insights which will be discussed in the next section.

5 Discussion

There was some progress on the purpose of this project, as described in the result section above. It is only vaguely visible if you were to play the game, as a lot of the components are missing, but as mentioned before the structure of the program became fairly solid. However, during the development of the game, the following information was discovered which might prove useful in continued development of the product, or in a new study along the same lines.

5.1 The Program Structure

The structure of the program is explained fairly thoroughly in earlier parts of this report and we will therefore only mention some of its advantages and disadvantages. One of the greatest issues of the program structure was the time taken before it was solidified, which led to some discrepancy between different parts. This discrepancy consisted of some parts trying to access information they were not supposed to have, and some of the parts having functionality that was intended to be in another part. As this was not noticed until a late stage of the production we had to work around it, which led to some very time-consuming issues when the separate NPC code was to be merged with the overall model.

The benefit of the structure is one that is mentioned earlier; it is fairly easy to scale and also very easy to debug since every part has its own duties and a problem within a specific section can therefore be easily traced back to its source. It could however have been improved significantly by adding a simple well-defined blueprint for the NPC, since that would remove the need for some of the workarounds which have been necessary due to coordination difficulties.

Something very noticeable with this kind of structure is that the complexity of somewhat basic operations increases quite a bit. This is usually a minor downside compared to the benefit of easy scaling and debugging, but in a somewhat small and focused project such as this, it can actually have a negative effect on the workload. To make the absolute most of the NPCs a program structure could even be designed after them, rather than fitting the NPCs into an already set one.

The game and the characters including the player look as intended and have not taken longer time than expected to implement. In the beginning of the project we made clear that the visuals should not be of a high priority but we also made clear that we wanted the game to look good. As the project progressed so did the graphics and in the end we reached a result we were happy with.

5.2 The PAD model

This model is easy to work with in a mathematical sense. Its axes are continuous and meant to be interconnected - particular meanings are assigned not only to the axes but also to the tuples[6] describing particular points - so that any function known from three-

[6] a finite ordered list of elements

dimensional geometry may be applied to it. This may be contrasted with alternatives such as the Big Five[7] model, which has five separate axes with little guidance on what effect different combinations of scores might have. Another useful thing about the PAD model is that it defines character as an average of moods, so that the same data structure can be used for both. This is also different from the Big Five model, which is intended solely to describe character.

In a linguistic sense, however, this model was a little trickier to handle. In particular, finding appropriate English descriptions for values where one of the axes has a value near zero was challenging. For example, what does one call the middle of a scale with "disdainful" at one end and "relaxed" at the other? More examples can be found in Appendix C. Also, some important distinctions, such as the one between platonic and romantic love, would not quite fit into the model. This work suggests that the model may not be quite expressive enough for building a really believable NPC. However, it still has more ready-made information attached to it than others, such as the Big Five model, so that it is a fairly good start for a project which does not include novel psychological research.

5.3 Similarity of the IThoughts to interlingual machine translation.

After the coding part of the project was concluded, it was noticed that the final shape of the IThought system curiously resembles interlingual machine translation (interlingual MT).
In interlingual MT the text to be translated is first transformed into a language-independent abstract representation of its meaning, a so-called interlingua. This interlingua is then translated into the target language(15). The similarity with the system developed in this project is obvious; essentially this system is interlingual MT without a source language. We speculate that this method and NPC AI might benefit from each other; currently known approaches in interlingua production might be re-used for NPCs while the obvious necessity of representing more and more information in machine-readable form for the benefit of AI could give a new impetus to interlingual MT research.

5.4 Data structures and methods for storing and retrieving IThoughts

The most important task for the data structure is to allow for retrieving information about a particular topic. For example, if an NPC is asked "Where is Mary?" it needs to retrieve all relevant information on Mary's whereabouts. In the current implementation the relevant information is defined as information with the same structure as the question, but really it should retrieve also information which is nested somewhere in a representation of somebody else's world: "Sarah thinks that Mary is…"

The difficulty with this is to compute a piece of information's relevance as an answer. While one is only dealing with structurally matching information, it is possible to use the certainty value for this purpose; however, this becomes difficult when you start including nested information: if you are 100% certain that Sarah thinks with 50% certainty that Mary is in the kitchen, you want your own 90% certainty that she is in the living room to

[7] for more information, see https://en.wikipedia.org/wiki/Big_Five_personality_traits

come first. On the other hand, if you have only a vague idea that Mary might possibly be in the living room, then it would make more sense to advance your beliefs about Sarah's opinion.

Compounding the problem of relevance, regardless of nested information, is time: If you are 50% certain that Mary was in the living room five minutes ago, then that is more relevant than your 100% certainty that she was in the bedroom fifteen minutes ago, and the information, with any certainty, that she was in the kitchen yesterday is so irrelevant as to not be any kind of answer. This project has not so far arrived at any sensible method of calculating relevance.

Inventing a suitable data structure for storing information is also difficult. When only the NPC's own view of the world is considered, this would be simple - just look in the right place and everything will be collected there. The difficulty lies in conveniently representing models of other people's worlds.

The first idea was to collect in the NPC's brain different data structures to capture people's whereabouts, the NPC's opinion about these people and so on, and then for each person the NPC knows a similar data structure, this time filled in with what the NPC believes to be inside the other person's brain. This was discarded early because in that case an extra data structure would need to be built to represent any given piece of information's content and nature - this because the NPCs need to be able to communicate isolated pieces of information instead of disgorging the whole of their information every time they talk.
An approriate solution for projcts which do not include much manipulation of information would therefore be to store information as a list of this extra data structure directly, without using an underlying more compact data structure.

The actual implementation of the project is a sort of hybrid of the two structures presented above, where some information pertaining to an NPC's own model of the world was placed separately with a view to making it easier for the programmer to access. However, from the perspective of programming effort and legibility, this technique turned out to make things more complicated to write and to understand.

5.5 Expressiveness of placeholders, null values and certainty

Although this system, using only two reserved values for placeholder and null respectively, usually works fairly well, there are situations where it is not sufficiently expressive. In particular, null values can clash when null in the sense of "there is no value here" is a valid value. That is not a problem with information such as whereabouts as it is not possible for a person or thing to be nowhere. However, it could be a problem when modelling information such as "Alice's murderer", because it might perfectly reasonable to think that Alice was not murdered at all and consequently has no murderer. In this case, one would have needed to introduce a second placeholder value.

Reusing the placeholder value as a question marker is also problematic, because it means that the ability to convey the information "Sarah knows where Mary is" is lost. As a placeholder value currently makes the program interpret the IThought as a question, this sentence would instead be interpreted as "Where does Sarah think Mary is?".

5.6 Making inferences from information

In the actual implementation, NPCs' reasoning powers are confined to noting that if somebody comes through the door to a room, they must previously have been in that room (and the reverse). However, there are many more inferences that could be drawn from data. For example, if an NPC sees something happen, it could infer that everybody else in the room also sees it. Further, if the NPC is informed of something, it could be led to believe not only that the speaker believes what they are saying, but also that the thing which they are saying is true. In the implementation, NPCs only make use of this idea when told background or "flavour" information such as "we are snowed in".

5.7 Advantages and disadvantages of only responding to the last statement

The obvious advantage, as mentioned above, of considering only the last statement made in calculating a response is that it is easy to do and requires less knowledge of the normal dynamic of conversations. However, there are also disadvantages. These have been largely glossed over by the game mechanics but are still worth considering if the game is to be extended, especially with regard to NPC-NPC conversation. For one thing, care needs to be taken to avoid endless loops, such as two NPCs eternally greeting each other. For another, it takes away the possibility of making a sensible response to a nonsensical answer: For example, if one NPC says "Where is Betty?" and gets the answer "We are snowed in", then it would be reasonable for that NPC to become annoyed. This reaction is not possible when only considering the last comment made. However, this may not be a very big problem under the assumption that NPCs always give sensible responses and only sensible responses are offered to the user.

5.8 Dialogue

The dialogues are, as it is now, not very believable. There is no real test to show the amount of believability in the dialogue, and there are no tests needed to realise that they are not. However the system is implemented to be able to expand the dialogues with more interesting things to talk about. It would be fairly simple to give the characters more options to answer a question with, instead of just having them answer with “I don’t know” or the answer to the question. Perhaps the character could lie or say that he or she believes something but is not completely sure.

The method we used to create our dialogue system has some advantages and disadvantages. The dialogues can easily be changed and modified, it is easy to steer any conversation in a certain direction and it does not limit the user too much in what he or she can say. There are some disadvantages, however; the fact that the input method is not free text severely limits the user, and risks the user losing immersion if he or she is not able to express him- or herself naturally. Another flaw in the dialogue tree is the amount of “clicks” it takes to say one line; a system like this can also break the user's suspension of disbelief.

The way we “translate” IThoughts into English also has some drawbacks. In this method, if one were to implement a new IThought, one also has to give that specific IThought a new translation. A much better way would be some code that can take an IThought and convert that IThought to English based on some variables accommodated in said IThought. How this system would look has not been in the scope of this project.

5.9 Evaluating believability

Believability, being subjective, is hard to define and harder to test for, a fact which has long been known by authors and critics of books and plays, which are the fore-runner of the modern role playing game in the sense that they all attempt to tell a story(2). The game produced in this project never got far enough to test in any meaningful way; however, this section suggests some methods for testing programs against our definition of believability.

Easily the most straightforward way of testing believability of any kind is simply asking the users. However, it has been suggested that it might be a better idea to record behaviour and then ask a third party to rate it, in order to minimize problems such as experimenter influence(5)(16). Also, one must decide what kinds of questions to ask - in particular, there is a difference between absolute ratings ("How believable is character A?") and relative ratings ("Which one of these is the most believable?")(5).

Another interesting approach is to analyze people's behaviour when interacting with the machine for clues; the idea is that a more believable machine will elicit more social dialogue, for example(6), or change the user's physiological responses(5)(17). However, this approach may be quite resource-intensive, given that not only are there many aspects to consider, but some of them, such as galvanic skin response, can only be measured accurately with special tools.

On a completely different note, an attempt was made in a paper by Lankoski and Björk to detail some components of believability with the aim of being able to verify objectively if they have been met (18). Some of these components which seem relevant to this project are

- awareness of surroundings - being aware of what is happening around one
- initiative - taking action without being explicitly prompted by an event
- own agenda - having goals of one's own
- emotional attachment - having emotional reactions to things or events
- contextual conversational responses - changing responses depending on context, such as whether a question has been asked already.
- goal-driven personal development - developing new goals when the old ones have been reached.

Though as has been mentioned the game is in a very early stage, evaluating this game with respect to these criteria can give some clues as to whether the development is headed in the right direction. It is found that while the NPCs exhibit some degree of awareness of their surroundings in the sense that they notice people entering or leaving the room, the other criteria have not been fulfilled. However, initiative, own agenda and emotional attachment were planned for although these components are not used in practice.

6 Possible extensions

During the work with this project many interesting ideas were discovered for which the time proved too short. However, they are included here in case somebody wants to extend the system or needs extra ideas for their own.

- One interesting idea is to use imitation learning to make characters seem more life-like(16). Though the authors of that paper are aiming for realism, it would be reasonably easy to adapt the system to character believability by training the system only on examples which test persons have found believable.

- Another extension which was originally planned for but never happened is the addition of a planner. For this, one could imagine integrating something along the lines of the planning system developed by Cohen and Perrault(13).

- The above-mentioned issue of calculating relevance from certainties and time might be valuable.

- Neural networks could be used for the decision making with a less predictable and more complex AI which could make the interaction with the user more rewarding. The problem of using neural network is the problem that arises when the network must be trained to accomplish a certain goal.

- Randomization could be used to a greater extent. For example, it would be possible to add to the code which generates multiple brains at once functionality which randomly selects one character to be the murderer.

- The inferring of information could be extended considerably, for example by teaching the NPCs to recognize implied shared truths. This fails completely in the following exchange:

  (Burt is standing in the same room as A and B unhurt).
  A: Do you know who murdered Burt?
  B: I don't know.

  If B had been able to infer from the question that A believes Burt to be dead, he would have given a considerably different answer.

- As mentioned before, there is a marked similarity between IThoughts and interlingual MT. As interlingual MT also involves translating from the source language to the interlingua, research in this area could be used as a starting point for the free-text parser omitted in this project.

7 Conclusion

After finishing the game the project group came to some conclusions regarding the NPCs, the AI and the project as a whole:

- Memory plays a large part in NPC believability. An AI has to remember events, locations and relations with other AIs in order to be able to display any manner of believable awareness. Each NPC needs a memory of its own so as not to seem omniscient to the user since that would drastically lower any kind of believability.

- The project also came to the conclusion that the AI has to at least give the impression that it reflects over events that occur and relations with other AIs. They have to seem as if they value events differently depending on who the event pertains to; for instance if a lover is badmouthed, the reaction of the NPC might be to act in their defence, but if it is a stranger, no reaction at all would be a more plausible reaction. It was found that the absence of such reflection was highly detrimental to believability.

- The decision-making of an NPC has to consider many different aspects to avoid seeming one-dimensional like the bots in a classic shooter, where only their current situation usually determines the next course of action. This project arrives at the conclusion that the relations between NPCs, their own personality, the current situation in the surroundings, the more global situation within the game, and the previous actions of the NPCs and player all need to be considered when creating a believable decision-making entity.

- The creation of a game and the implementation of believable NPCs along with a solid conversation system is a big task. Something that we realized during the process was how much bigger each of these subtasks were than anticipated. Separation of these tasks and specifying one or two as the purpose might give a future study a chance of reaching completion before the deadline instead of reaching semi-completion within all three.

- Using AI literature often presumes some prior knowledge about implementation of AI and therefore often only discusses the abstract concepts rather than implementation practices. Some prior experience about the basics of AI implementation is something we recommend any pursuers of this type of project to have.

- Combining the different parts of the AI, model and conversations proved to be a daunting task; however, it is one that could have been made easier through more specific outlining and structuring of the whole end-product at an early stage.

8 References

(1) Deryugina O. Chatterbots. Scientific and Technical Information Processing. Apr 2010. 37.2:143-147.

(2) Mateas M. An Oz-Centric Review of Interactive Drama and Believable Agents. Volume 1600 of the series Lecture Notes in Computer Science: 297-328.

(3) Fairclough C, Fagan M, Mac Namee B, Cunningham P. Research directions for AI in computer games [Internet]. 2001 [Cited 2016-02-05]. Available from: http://edepositireland.ie/bitstream/handle/2262/13098/TCD-CS-2001-29.pdf?sequence=1&isAllowed=y

(4) Arkowitz A, Lilienfeld S. Why Science Tells Us Not to Rely on Eyewitness Accounts. Scientific American Mind [Internet]. 1 January 2010 [Cited 2016-05-31]. Available from: http://www.scientificamerican.com/article/do-the-eyes-have-it/

(5) Togelius J, Yannakakis G, Karakovskiy S, Shaker N. “Assessing Believability”. in Hingston (Ed.) Believable Bots, pp. 215-230, 2012. Springer Berlin Heidelberg.

(6) Tencé F, Buche C, De Loor P, Marc O. The challenge of believability in video games: definitions, agents' models and imitation learning. [Internet] Cited 2016-05-10. Available from: http://www.enib.fr/~buche/article/GAMEON_10.pdf

(7) Simmons R, Makatchev M, Kirby R, Lee M, Fanaswala I. Believable Robot Characters. AI Magazine. Winter 2011.

(8) Orkin J. Three states and a plan: The A.I. of F.E.A.R.. Game Developer Conference 2006, [Internet] Cited 2016-05-10. Available from: http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf

(9) AI Programming Overview [Internet] Valve developer community, Apri 2008, Cited 2016-04-20, Available from: https://developer.valvesoftware.com/wiki/AI_Programming_Overview

(10) Lester P. A* Pathfinding for Beginners. [Internet] 2005, cited 2015-04-11, Available from: http://www.policyalmanac.org/games/aStarTutorial.htm

(11) Mehrabian A. Pleasure-Arousal-Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament. Current Psychology: Developmental Learning Personality Social. Winter, 1996, Vol.14, No. 4, 261-292.

(12) Frith C, Frith U. Theory of mind. Current Biology, Volume 15 Issue 17, [Internet] sept 2005, Cited 2016-05-10. Available from: http://www.sciencedirect.com/science/article/pii/S0960982205009607

(13) Cohen P, Perrault C. Elements of a Plan-Based Theory of Speech Acts. Communications in Multiagent Systems, LNAI 2650, pp. 1-36, 2003.

(14) Ellison B. Defining Dialogue systems. [Internet] Cited 2016-05-10, Available from: http://www.gamasutra.com/view/feature/132116/defining_dialogue_systems.php

(15)
Hutchins J, Somers H. An introduction to machine translation. [Internet] London: Academy press 1992. [ISBN: 0-12-362830-X]. Cited 2016-05-29. Available from: http://www.hutchinsweb.me.uk/IntroMT-TOC.htm

(16) Gorman B, Thurau C, Bauckhage C, Humphrys M. Believability Testing and Bayesian Imitation in Interactive Computer Games. S. Nolfi et al. (Eds.): SAB 2006, LNAI 4095, pp. 655–666, 2006. Springer-Verlag Berlin Heidelberg 2006

(17) Ghaznavi M.Using psychophysiological techniques to measure user experience with entertainment technologies. [Internet] Affectivegaming. April 2013, Cited 2016-05-29. Available from: http://www.affectivegaming.info/csci5550/hci-grad-2013/milad/paper-using-psychophysiological-techniques-to-measure-user-experience-with-entertainment-technologies/

(18) Lankoski P, Björk S. Gameplay design patterns for belivable non-player Characters. Situated Play, Proceedings of DiGRA 2007 Conference. [Internet] cited 2016-05-29. Available from: http://homes.lmc.gatech.edu/~cpearce3/DiGRA07/Proceedings/055.pdf

Appendix A: The proposed user test

The point of the user test is to find out how believable the NPCs appear to the user. This is not a fully-fledged Turing test but more of a survey that gives us an idea how well we reached our goals.

This test is intended to be administered to a fairly small number of users - maybe half a dozen - and the results analyzed in depth. It is intended to proceed as follows. First the supervisor explains what the game is about and what the goal is. The user plays the game and the supervisor helps by suggesting what to do when the user gets stuck.

After the user has played about 10 minutes the test is ended. The user is then interviewed by the supervisor. The following questions are asked:

- How did you think the NPCs appeared?
- Did you find the NPCs' responses to be accurate?
- Did you find anything strange?
- Did you notice that the NPCs remembered what you did in the game?
- Did you notice that the NPCs …?
- How plausible do you think the NPCs were on a scale from 1-10?
- Can you tell me which character was controlled by the assistant? (trick question)

Appendix B: Use cases

Maps, movement and scenery

Name: User Movement
Description: Move the player in the view, with keyboard controls
Purpose: The user should be able to move the player around.
Prio(1-5 (low-high)): 5
Flow:

1. User inputs a direction in which they want to move.
2. System checks if movement is valid.
3. System moves the player if movement is valid.

Name: Interact with objects
Description: Player interacts with non-NPC
Purpose: The player should be able to interact with the environment.
Prio(1-5 (low-high)): 5
Flow:

1. User presses the interact button.
2. If the object/tile? in front of the player is interactable it is interacted with.

Name: NPC interaction with objects
Description: lets an NPC interact with interactable objects
Purpose: The NPC should be able to interact with objects.
Prio(1-5 (low-high)): 2
Flow:

1. NPC moves to a interactable object.
2. NPC interacts with object.

Name: Scene change
Description: Change the scene(room) when player leaves the current room
Purpose: Only the room the player is currently inside should be visible.
Prio(1-5 (low-high)): 5
Flow:

1. User uses Use Case "*Character change room*"
2. System displays the room the user moved to.

Name: Character change room
Description: Characters move from one room to another
Purpose: Let the characters move around in the building
Prio(1-5 (low-high)): 5
Flow:

1. A character interacts with a door.
2. System moved the character to the tile which the door is connected to.

Name: Display Map
Description: Castle map

Purpose: to avoid getting lost, also to avoid making adjoining rooms too grotesquely out of proportion. Further, might be used to make NPC understand consequences of movements (guess at where somebody is going to be by what door they are going out of)
Prio(1-5 (low-high)):3
Flow:
1. Press m to show map.
2. Press m again to hide map.

Name: NPC movement
Description: NPC moves from one location to another
Purpose: NPCs should be able to move around the castle.
Prio(1-5 (low-high)): 4
Flow:
1. NPC AI decides to go to a specific location.
2. A pathfinder finds a route to the location.
3. NPC moves along the path.

Interaction and dialogue

Name: NPC choosing replies
Description: NPC takes any necessary input (user response, emotion, class…) and calculates a response
Purpose:
Prio(1-5 (low-high)):4
Flow:
1. The NPC receives a speech act object.
2. The statement and the opinion of the character in question are evaluated.
3. The NPC choose an appropriate reply.

Name: initiate Player-NPC dialogue
Pre-condition: Interact with NPC (or NPC interacts with player?)
Description: Shows, and start the conversation with a NPC
Purpose: to talk to a specific npc
Prio(1-5 (low-high)):5
Flow:
1. Dialogue box appears at bottom of screen, containing either a player multiple-choice or a comment from the NPC.

Name: NPC emotions
Description: NPC calculates emotions from temperament, current mood, action, etc
Purpose: To make the NPCs seem credible.
Prio(1-5 (low-high)):4
Flow:
1. Either NPC is prompted by system, or an event happens
2. NPC recalculates its emotions

Name: NPC interacts with other NPC

Description: Two NPCs interact with each other.
Purpose: NPCs should be able to share information without the player interacting with them.
Prio(1-5 (low-high)):4
Flow:
1. NPCs send speech act objects to each other

Name: Group conversations
Description: Conversations around dinner table and so on with player and multiple NPCs
Purpose:
Prio(1-5 (low-high)): 1
Flow:
1. NPC sends same speech act object to all NPCs in the same room

Name: Whispering
Description: NPCs or NPCs and player having a conversation which can't be heard by others in the same room. Should be the default.
Prerequisite: Maybe talking while turned towards a person on the next square?
Purpose: People exchanging information can be a clue
Prio(1-5 (low-high)):3
Flow:
1. NPC sends speech act object only to person they are facing

Name: Reply generation for player
Description: Calculate three different replies for the player to choose between
Purpose: To limit the available responses by the player. Avoids the need to build a speech parser.
Prio(1-5 (low-high)):4
Flow:
1. System indicates it wants the player to say something
2. Reply generator creates sentences

Name: NPC thoughts
Description: NPC reactions to other people's actions.
Purpose: People's recollections of things provide clues to the mystery as well as credibility.
Prio(1-5 (low-high)):4
Flow:
1. Something happens (somebody does something) in the room the NPC is in
2. NPC may choose to remember action and/or generate feelings in response to action.

Name: generate NPC
Description: A NPC is generated with (random) traits, minding family relationships.
Purpose: To easy be able to generate multiple NPCs
Prio(1-5(low-high)): 4
Flow:

1. Game is started
2. NPC is generated

Name: Set murder trait
Description: (Randomly) sets a single NPC to Murderer with motive.
Purpose: To make sure there is a murderer in the game.
Prio(1-5(low-high)): 4
Flow:
1. Game is started
2. NPC generation is influenced such that one NPC becomes likely to commit a murder in the near future.

Appendix C: The meanings of the PAD model

The list below shows the meanings given to each octant in the PAD model. The plus and minus signs represent the sign of that axis in the model. The temperament descriptions are those proposed by the creator of the PAD model, Albert Mehrabian. The words for feelings and opinions were derived from these in a very unscientific manner, partly by using a thesaurus and partly by attempting to answer - very subjectively - the question "If somebody or something makes me feel [insert temperament here], then what feelings do I have about this person or thing?"

The words used to described emotions differ from the ones describing temperament despite temperament being defined as an average of emotions because the temperament descriptors were sometimes not quite intuitive to non-psychologists.

PAD	**Temperament**	**Opinion**	**Emotion**
(+P+A+D)	Exuberant	Loves	Cheerful
(-P-A-D)	Bored	Not interested in	Bored
(+P+A-D)	Dependent (needy)	Relies on	Wants somebody to rely on
(-P-A+D)	Disdainful	Dislikes	Scornful
(+P-A+D)	Relaxed	Likes	Relaxed
(-P+A-D)	Anxious	Scared of	Anxious
(+P-A-D)	Docile	Wants to obey	Cooperative
(-P+A+D)	Hostile	Hates	Angry

An attempt was also made to give matching definitions to the various quarters of the cube to be used when one of the axes is near zero. However, this attempt illustrates mostly the difficulty of deriving such definitions from the definitions of the octants above. Mehrabian himself has only very few and incomplete comments about these values, although the comments he does make could be quite inspirational. The information he gives includes a connection between sensitivity to rejection with quarter +A-D and a connection between physical activity and quarter +P+D.

PAD	**Temperament**
(+P+A)	Empathy
(+P-A)	Emotional stability
(+P+D)	Extroversion
(+P-D)	Lack of initiative
(-P+A)	Irritability
(-P-A)	Dismissive
(-P+D)	Disdainful/hostile?
(-P-D)	Lonely
(+A+D)	Impulsive
(+A-D)	Impotent
(-A+D)	Disdainful/Relaxed?
(-A-D)	Bored/Docile?