MALMÖ HÖGSKOLA

Teknik och samhälle
Datavetenskap

Examensarbete
15 högskolepoäng, grundnivå

Behavior Based Artificial Intelligence in a Village Environment

Beteendebaserad Artificiell Intelligens i en bymiljö

Tim Lindstam
Anton Svensson

Examen: Kandidatexamen 180 hp
Huvudämne: Datavetenskap
Program: Spelutveckling
Datum för slutseminarium: 2017-06-01

Handledare: José María Font
Examinator: Steve Dahlskog
Template: www.emena.org/
files/springerformat.doc

Behavior Based Artificial Intelligence in a Village Environment

Tim Lindstam, Anton Svensson

Game Development Program, Computer Science and Media Technology, Malmö University, Sweden

Abstract. Autonomous agents, also known as AI agents, are staples in modern video games. They take a lot of roles, everything from being quest-givers in roleplaying games, to opposing forces in action- and shooter games. Crafting an AI that is not only easy to create, but also retains humanlike and believable behavior, has always represented a challenge to the development industry, and has in several cases ended up with open world games using AI systems that limit the AI agents to simple moving patterns. In this thesis, a form of AI systems more commonly used in simulation games such as *The Sims* video game series, are taken and implemented in an environment that could possibly be seen in an open world game. After the implementation, a set of tests were performed on a group of testers which resulted in the insight that a majority of the testers, when asked to compare their experience to other games, found this implementation to feel more lifelike and realistic.

Keywords: AI, game development, game AI, behavior tree

1 Introduction

The world of games is not only inhabited by the players of video games, but also by autonomous agents. In certain games, such as in shooter games, these agents communicate with each other, give each other orders, and will also attempt to outsmart the players of the game. This gives the player of the video game the perception that these enemy soldiers are a force to be reckoned with, something with the potential to not only shoot you down, but also to outsmart them with different tactics (Schwab 2008).

In the Black & White series (Lionhead 2003), a villager can be assigned a job by the player, acting god. When the villager has been assigned a job, it will perform that job until it dies or is reassigned a new job. Some of the jobs featured in the game were farmer, woodcutter, breeder and builder. If a villager is not assigned a job, it will wander around the village to simulate a living village. Besides their jobs, they also have certain needs. They want food to eat and a house to sleep in, otherwise they will eventually starve or freeze to death by sleeping outside. In the original Black & White, the village storage was the midway point of the village, since this building allowed the player to see what the village desires and needs are. According to Yildirim and Sindre Berg Stene (2008), the artificial intelligence (AI) of the Black & White series shows a more realistic AI than what is commonly seen in games, to the point where the AI can learn things on its own. For example, the creatures in the game, after some time, will learn how to greet its master and know to how to correctly act based on its beliefs and perceptions (Yildirim and Sindre Berg Stene 2008).

Other games require nothing more than a character which wanders aimlessly around in the game world, not doing anything before the player interacts with them (Schwab 2008). The latter of these is what is usually found in roleplaying games (RPG). RPG games commonly allows a single player to explore a large world filled with towns, villagers and foes to take on. The player usually starts out weak and then slowly grows in power (Schwab 2008). An example of such a game is the Dragon Quest (Square Enix 1986) series by Square Enix, formerly Enix. In these games, the player is mostly free to explore the world, it is continents and its settlements as they themselves desire. The settlements in these games vary in size, sometimes smaller villages, sometimes larger castle towns, but common for these are its static inhabitants. The inhabitants do not fill a function when it's not interacting with the player. Some of them will walk around, some of them will not. The same is also true for many other similar games, and not isolated to the Dragon Quest series. Game Freak's Pokémon series (Game Freak 1996), as well as a majority of Square's Final Fantasy (Square 1987), and the EarthBound games (HAL Laboratory 1989) all use this static villager system.

Compare this to the Sims (Maxis 2000) series by Maxis. The *Sims* series is one of many "god games" which puts the players in control of the lives of several artificial agents (Schwab 2008). In the Sims, the player is "a god", controlling practically every aspect of the lives of their "Sims" characters inhabiting the world. These "Sims" are human lookalikes, and their lives and society is modeled after our own, allowing players to play with an 'advanced' dollhouse. These Sims can also, with varying degrees of success, make their own decisions depending on their needs. The status of "I am sad because I am hungry" will eventually lead to the character to cook some food if the player does not order them to do so. This semi-lifelike simulation is miles ahead of what is being presented in the previously mentioned Dragon Quest series, and this issue is still prevalent today. In 2016, Pokémon: Sun and Moon (Nintendo 2016) and Final Fantasy XV (Square Enix 2016) were released, and both games feature a static AI compared to the AI seen in the previously mentioned Sims series.

1.1 Purpose

We intend to implement AI-systems more commonly seen in simulation games, into environments where they are not as commonly seen or used.

This system will include smart objects and a behavior tree planner, which together will be able to simulate villagers in a more realistic way compared to the more commonly used 'static' inactive AI's and AI's using a scheduler for timekeeping commonly seen in modern computer role playing games. By using need-driven behavior, smart terrain and a planner, the villagers should be able to become more

realistic. The villagers will be able to plan their day in advance instead of the simply walking back and forth.
By also implementing shared resources, such as wheat, lumber and bread, all of which the villagers need to fulfill certain actions and needs, the villagers will be able to accomplish a more realistic goal and appear to be more real to the players than their static or scheduled variants. These teqniques will be introduced in the Introduction section.

Following this implementation, we will test the software with several testers and evaluate the overall experience of realism in the environment we have created. These test players will be exposed to two versions of our test software, one which feature agents directly controlled by a test supervisor, and one controlled entirely by the behavior based AI we have proposed.

After the session, the test player will be asked to answer a questionnaire. This data will serve as the basis of our results. The questionnaire will be different depending on what test the tester is subjected to. The intention is to mix up what is told/done between tests. For example, some testers will be asked to find the human controlled agent, when in fact, all agents are controlled by the AI. The outcome of these questionnaires will be explored and discussed towards the end of this thesis.

Related work has been done in this field. In one related work, the authors made a similar approach by used smart terrain and needs, but the authors try to generate behavior for the AI using cycling schedule as a reference (Sahlin and Olsson 2005). In this thesis, a similar approach will be conducted by using “semi smart object”, behavior tree and a planer.

1.2 Behavior tree

A Behavior tree is a tree of hierarchical nodes which control the decision making of an Artificial Intelligence (AI) entity (Millington et. al. 2009). The branches of a Behavior tree have several types of utilities which controls the decision-making path as it navigates down the tree. The leaves of the tree are the commands the AI agent can perform. This is a major difference from other types of AI systems.

These behavior trees can be extremely complex and feature a vast number of branches which can be called sub-trees in their own rights, specializing functions. This allows for creating smaller behavior trees which, chained together, creates a very complex AI system and results in very convincing AI behavior.

A behavior tree consists out of several types of nodes. These can normally return one of the three common return-statuses: success, failure and running.

The first two return types will notify the parent-node if the operation was a success or a failure. The third will only notify its parent-node that the current state is not yet determined and an operation is still in progress. This will continue each system-update until the node has determined its state.

One of the first big games to use a variant of a Behavior tree, is Halo 2 (Bungie 2004). In *Halo 2*, the AI's movement are all divided into smaller sequences, which, depending on the circumstances, can be triggered (Gamasutra 2005). For example, in Halo 2, if the player kills the strongest enemy in a group, the smaller, weaker enemies will trigger a retreat sequence.

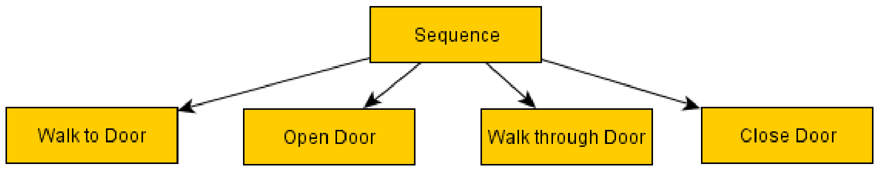


Figure 1 - Behavior tree example by Chris Simpson published on Gamasutra (Gamasutra 2014).

1.3 Goal oriented planner

A goal oriented planner (planner) is a system that allows an agent to plan a sequence of actions to satisfy a goal. The job of a planner is to take the information it has concerning its agent and combined it with information about its environment (Millington et. al. 2009). With the additional information, it will decide which actions are the most beneficial for the agent to execute and satisfy its current goal (Millington et. al. 2009). In our case, the planner's task is to process the needs and environment, and plan action for the agent to take. Even though the same goal is supplied for multiple agents, their action to complete the goal can be different, some might take an altering way of movement or take the fastest way. This will make the AI more dynamic and realistic. *The Sims* (Maxis 2000) series use this type of goal oriented planner to plan several actions with the help of smart terrain objects (Diller et. al. 2006).

1.4 Smart terrain

This is a technique where the logic and data is stored inside the objects scattered throughout the world. Such object is called smart objects. These so-called Smart terrains contain all data necessary to exist within the world, but also how the different agents and the player can interact with it (Schwab 2008). The benefits of using this technique is that the logic of the object reside inside the object, eliminating the need to rewrite the game logic when a new object will be put into the game hence new objects can be easily be added into the game. Due to this set up, tools can be created, which can be used to create more smart objects (Diller et. al. 2006). The *Sims* (Maxis 2000) series shows off this technique, as it is used throughout the whole series. Every object that the player can buy is a smart object that can easily be placed in the game world. The smart objects will continuously broadcast messages, which the agents (*Sims*) receive, telling them that they can perform a certain task here, along with the location of the object itself. Usually these "tasks" involve satisfying a certain need for the agents, such as how beds will satisfy the sleeping need. They can also carry with them costs as well. The fridge, for example, can satisfy the hunger need, but it will cost the agent in game currency to use it.

Our implementation of smart objects will not include all above mentioned aspects of smart objects, and thus differ slightly in use. This will be mentioned in Section 1.6.

1.5 Research question

To test and implement our proposed solution described so far, we have decided to answer the following questions:

1. To what degree can the behavior system be implemented in a village-setting in a role-playing game where you do not typically see them?
2. To what degree does this affect the perceived realism by players when compared to the classic approaches?

1.6 Constraints and scope

This thesis will be limited to the AI's already mentioned, smart objects, need, and behavior trees, as well as the evaluation of these. It will not attempt to evaluate any further AI variations, or attempt to implement procedural content generation, even though mentioned in referenced literature.

We will not discuss or attempt to explain what happens if no smart objects exist for the AI to deal with.

There are roleplaying games like the Elder Scrolls (Bethesda Game Studios 2002) and Fallout (Bethesda Game Studios 2004) Series which employs a different technique in order to achieve a greater amount of realism in their villager AI, called Cyclic scheduling (Zhao and Szafron 2014). However, as these do not employ Behavior trees or a goal oriented action planner, we will not focus on these in our thesis.

Our implementation of smart object will contain all the aspect of smart objects, except the broadcasting, which we decided not to implement. Our reasoning for this is that we consider this unnecessary for our artifact, due to the other AI systems that contains the same function, such as the Behavior tree.

1.7 Expected results

We expect to find that the living organic village will be more interesting or appealing than a static village commonly seen in some games. That said, what we or the tester believe is more "interesting" or "appealing" might not mean it's an inherently better solution for all implementations and games.

We expect the results of the questionnaire to vary greatly. For example, we believe that non-gamers might find the living village with villagers living out their lives more interesting than a village where villagers simply stand around, but we expect that the "core gaming" crowds might find them distracting.

2. Method

This section of the thesis will be presenting and discussing our method for answering the research questions in section 1.3 of this thesis.

The method chosen to solve this problem is the *Design Science* method. The primary ambition of design science is to improve a specific area with a known problem by creating artifacts, thus this fits our intentions. Design Science is explored in depth in Peffers et al. (Peffers et. al. 2009).

2.1 Research process

The research process follows these steps:

1. **Problem identification and motivation:** This step involves defining the research problem(s) to justify the value of a solution. This step is important, because it will motivate both the researchers to pursue a solution, and the audience to accept the results, as well as understanding the reasoning of the researchers.
2. **Define the objectives for a solution:** This step is about defining the objectives by drawing upon the problems and knowledge from the previous step, ensuring that the objectives are possible and feasible.
3. **Design and development:** Once the objectives are defined, the artifact creation can begin. Resources required for moving between objective defining and design and development is a knowledge of the theories that are needed to support the artifact.
4. **Demonstration:** The demonstration part is where the artifact is being tested in one way or another to demonstrate its usefulness to solve the problem. This can be done in several ways, such as simulations, case studies and similar activities.
5. **Evaluation:** To evaluate the produced artifact, observations and measurements are necessary. Depending on the nature of the given artifact, the evaluation may vary between projects. There may in some projects be additional simulations, while satisfaction surveys are totally reasonable for other projects.
6. **Communication:** The communication is the concluding section, where the artifact and its utilities, novelties, design rigor is and effectiveness are communicated to the relevant audiences.

2.2 Implementation of the research methodology

This given research process was used by us, and translated into the following process:

1.Problem identification and motivation: Given the games presented in Section 1, the problem as we have defined it is the relative static behavior, which impacts the general perception of realism for a player.

2. Define the objectives for a solution: We decided upon the objectives of creating a villager AI with a higher amount of perceived realism, as well as upon implementing smart terrain, a planner, and a need driven behavior based upon the AI used in the Sims, to create what we believe is a more lifelike villager ecosystem.

3. Design and development: The AI and the "test-platform", a small game with the intention of testing the AI, was developed using the Monogame framework for Visual Studio C#. The "test-platform" also contains a village, which will be used for evaluating the artifact.

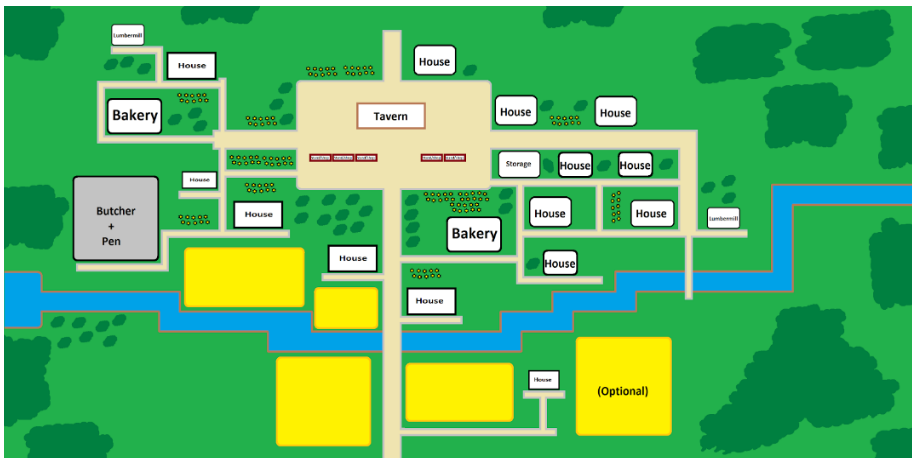


Figure 2 - Planned village layout during the initial phases.

4. Demonstration: To demonstrate the effectiveness of our artifact, several tests were executed with different testers. Testers will be given two tasks, either spotting the test supervisor, a human supervisor controlling a character in the village, or simply observing. After every test, the testers will be asked to fill in a questionnaire.

5. Evaluation: As mentioned previously, questionnaires were used to evaluate the tests performed in the 4th phase. Two "sets" of questionnaires were given to the testers. The first set are different questionnaires concerning the evaluation of the tests. The second set is a big questionnaire with questions allowing the tester to evaluate the AI. By using different questionnaires, we hoped to receive different type of data, some specific to the tests, and some about their general opinions about this AI. Questionnaires are explained in detail at Section 4.

6. Communication: The communication phase is the final part of the project, and will in this thesis reflect the last three Sections, 5, 6 and 7 of this thesis. In these sections, the results are presented and analyzed, as well as discussed.

The test sessions saw testers trying to figure out whether the agents were controlled by a human person, or if it was controlled by the computer. The tester was controlling a character with a limited set of possible actions, which will be able to move around the village. A more in-depth description at Section 3.3.

2.3 Method discussion

This Section will discuss the method presented.

2.3.1 Questionnaire

We created a questionnaire based on the Likert scale (Tullis and Bill 2008). This scale features questions along with a 5-mark scale, where a “1” means to “strongly disagree”, a “3” means to “neither like or dislike”, and a “5” means to “strongly agree”. The questionnaire also featured many questions where the testers can voice their opinions. By mixing Likert scale questions as well as open write-in questions, we hope to receive a satisfying combination of usable evaluation data.

Our main reasoning behind choosing the Likert scale was its ease of use. The most key factor is to keep the amount of possible answers uneven to have room for at least one neutral answer. If one follows this rule, you could potentially bump up the amount of possible answers in the questionnaires to seven, if it makes sense for the matter, and that it is free from adverbs (Tullis and Bill 2008).

There are also valid arguments for using semantic differential scales instead of Likert scales, however, these require opposite alternatives as questions, which could be problematic. We think the Likert variation of writing questions is more straightforward (Tullis and Bill 2008).

A potential issue, which we must consider when letting users take the questionnaire, is leaving them alone when they answer the given questions. Loss of anonymity could potentially contaminate the data. According to Tullis and Bill, there exists an amount of “social desirability bias”, which is when respondents give answers that they believe will make them look better, or not to disappoint the evaluator (Tullis and Bill 2008). This could potentially change the entire outcome of the project.

3. Solution

This section will detail the solution we have developed to answer the questions stated in Section 1.3.

3.1 Engine and framework

The software was written in the Monogame framework (Monogame 2009), an open source continuation of Microsoft XNA framework (XNA), which allows for the software development of game applications for several platforms.

This was used together with an open source tile editor called Tiled (Tiled 2017), which allows for easy creation of levels and worlds.

3.2 Software

The software we have proposed will allow the player to traverse and interact with a village (see Figure 3), and its inhabitants.

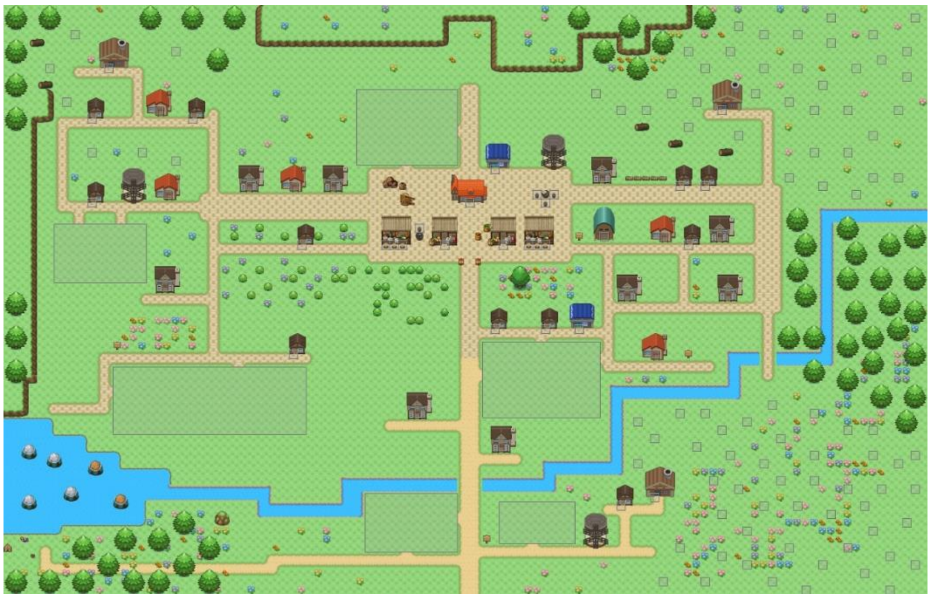


Figure 3 - The village designed for this thesis.

The village is inhabited by AI-agents created and specified by us. These AI-agents each fill one role in the village, such as being a baker, a traveler, a lumberjack or a farmer. They cannot occupy more than one role in the world, and they cannot change role at any later time. Upon creation, they will also "receive" a predefined house. This is house serves as this AI-agents home, and they will enter it when they want to sleep, or when if they have bought some food and want to eat at their home. Each AI-agent will also receive a random name.

The village will be built upon several designated areas, such as lumber cutting areas, bakeries, storages, farmlands, and houses "owned" and inhabited by individual non-player characters (NPC) or NPC families. The game will feature more than one of these. For example, there are two different areas where lumberjacks can cut down wood. They can cut wood at both places, but will be biased towards the one closest to their location.

The central square in the world features some vending stalls and a tavern where villagers can meet, socialize and buy food if they don't want bread from the baker. The tavern is the village's main socializing and eating hub, but only the occasional traveler will spend the night there.

The world features a day and night cycle, where one full 24-hour cycle consists of four real time minutes. Villagers will traverse the land and fulfill its needs based on their respective needs in the system.

3.3 Users and user input

The project overall consists of two roles; the supervisor(s), and the tester. The test supervisors, always two, have two main jobs. The first supervisor will help the tester getting started. This means that any questions will be answered by this person. This is the person that will also give the tester their "mission". The second supervisor will be either controlling or pretending to control an AI character. This supervisor will sit opposite to the tester. In certain tests, this person won't be introduced to the tester until after the test. The tester will perform the role of a visitor in the village. The tester will be tasked with different goals, explained in detail at Section 4.

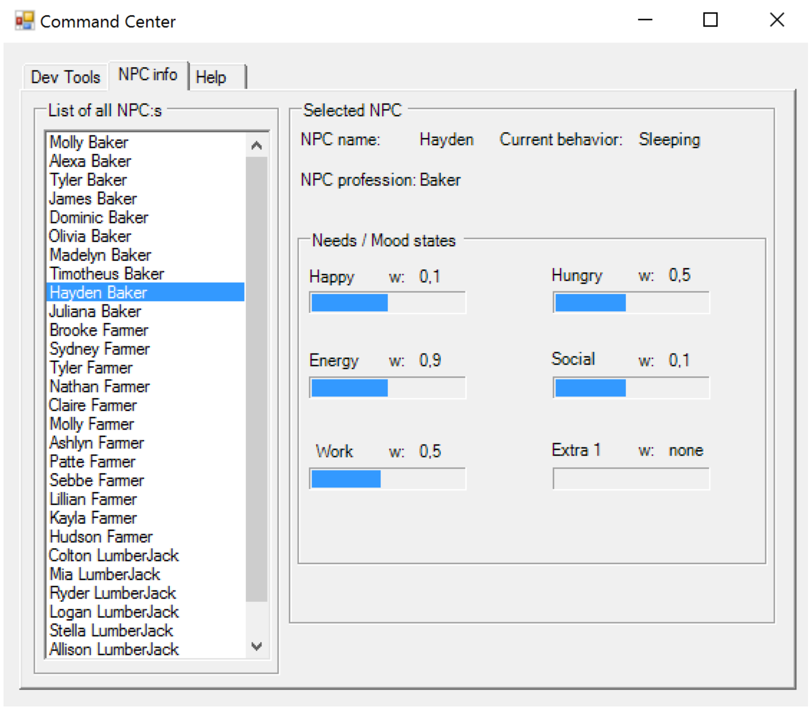
The game will feature two concrete different in-game modes:

The first mode is the spectator/traveler mode which the testers will be using. This mode will be simulating the experience of a real game. The user can walk around in the village, and socialize with the inhabitants and spectate their daily lives. In this mode, the testers will not be able to perform or complete any of the work tasks given to the workers in the village. We do not intend for the test users to plow the fields or cut down trees, only to spectate and socialize. In addition to walking around the village and talk with the inhabitants, the test user can also enter the tavern and eat some food. The test user can do this however many times they want. There will not be anything in the game forcing the test user to eat.

The mimic-mode is the mode that will be used by one of the test supervisors. The supervisor can walk around and mimic actions of the inhabitants. The mimic user can perform all tasks in such a way as to be perceived as an actual AI by the tester.

3.4 The Command center

The software will include a small command window (See Figure 4), which will be hidden from the test user.



List of all NPC:s

Molly Baker
Alexa Baker
Tyler Baker
James Baker
Dominic Baker
Olivia Baker
Madelyn Baker
Timotheus Baker
Hayden Baker
Juliana Baker
Brooke Farmer
Sydney Farmer
Tyler Farmer
Nathan Farmer
Claire Farmer
Molly Farmer
Ashlyn Farmer
Patte Farmer
Sebbe Farmer
Lillian Farmer
Kayla Farmer
Hudson Farmer
Colton LumberJack
Mia LumberJack
Ryder LumberJack
Logan LumberJack
Stella LumberJack
Allison LumberJack

Selected NPC

NPC name: Hayden Current behavior: Sleeping

NPC profession: Baker

Needs / Mood states

Happy w: 0,1

Hungry w: 0,5

Energy w: 0,9

Social w: 0,1

Work w: 0,5

Extra 1 w: none

Figure 4 - Command Center, part of the software, used to gain information about villagers.

This software will output in-depth information about selected AI-Agents, such as current amount of energy and how hungry they are now, or what action they are currently doing.

This not only serves as a good debugging tool during the earlier development stages, but will also help the test supervisors, make sure everything is working as intended and that the inhabitants are not being affected by having too many needs going unsatisfied, which could contaminate the testing and evaluation sessions.

3.5 Decision making

Needs

Needs represent the requirements and wants of the AI agents. Inspired by the "needs" featured in Maxis the *Sims* (Maxis 2000), the strength of these needs can increase or decrease depending on the actions of the agents. If an agent is too hungry, it will try to obtain and eat food. If it is bored, it will try to find someone to talk with.

To make this system more unique and different from more common implementations, the needs in our AI agents are weighed and randomized at start. This means that certain agents might prioritize eating lots of food, while other agents might desire to talk with people more often. This weight work in both ways, certain AI agents, for example, might desire not to interact with other AI agents because the weight makes it least prioritized sequence.

Once a sequence has been performed by an agent, its needs will change in separate ways, as can be seen in Table 1 below. Example, villager Simon's "Hunger-need" is very low. This means he has not eaten any food in a long time. The obvious action to remedy this situation is too eat. Eating will result in the "Hunger-need" value decreasing, meaning it will no longer be prioritized. Certain sequences, such as sleeping, also have a "negative" attribute. When an agent is sleeping, its "energy-need" will increase, meaning it won't try to sleep again too soon, but its "Hunger-need" will also decrease, meaning that the agent, after an extended period of sleeping, will try to get something to eat.

This system works well, but can be a bit complicated due to the abstract nature of certain needs, especially Work. If the "Work-need" is too low, the agent will attempt to work. While working, the "work-need" will increase, while a lot of needs will decrease.

Table 1: Need-Sequence relations.

Low need	Action necessary to increase need	Need value increasing	Need value decreasing
Hunger	Eating	Hunger	-
Energy	Sleeping	Energy	Hunger
Socialization (Loneliness)	Socialize	Loneliness, Work, happiness	-
Happiness	Wander around	Happiness	Energy, Hunger
Work	Working	Work	Energy, Hunger, Socialization, Happiness,

The game simulates 24 hours of in game time. To accurately simulate the needs over time, the needs will degrade over time, with different amounts depending on if it is currently daytime or nighttime. Table 2 represent this degradation.

Table 2: Degradations over time, this represents 24 hours of in-game time.

	Daytime degradation	Nighttime degradation
Hunger	High amount	Low amount
Energy	Very high amount	None
Socialization	High amount	None
Happiness	Low amount	High amount
Work	Low amount	None

Example: Agent Simon the farmers “Hunger-need” will degrade a lot during the day when he is active and out in the world, but it won’t degrade as much during the night. Meanwhile, his “Happiness-need” will degrade with a low amount during the day as he is constantly fulfilling tasks, like talking, working and eating. However, as he goes home for the evening and goes to bed, his “Happiness-need” will decrease a lot, as he is bored.

Smart objects

Interactive objects in the village, such as doors, trees, farm-plots, taverns and such, are all smart objects. These objects can be grouped up together, thus making it easy for us to use it when pathfinding. The normal implementation of smart objects also includes broadcasts as well, however, we chose not to implement this. We discuss this in the solution discussion, in Section 3.7.

Planner

The agent’s planner will evaluate its current needs statuses and plan three different sequences that will satisfy its current needs. A new evaluation will not occur until all the behaviors-sequences have been executed or one of its needs reaches a critical point that forces the agent to re-priorities its planned behaviors.
The algorithm that agents use to decide its behavior is as follows.
The need (N) describes as follow

$$N = w * f * v$$

w = The “needs” weight (defined by the agents).

f = A world factor that can increase the weight-amount depending on the environment. The default value is 1, so if no factor is affecting the agent, nothing will change. An example, when it is nighttime the **f** factor will increase three times for the sleeping need compared to the normal daytime need, “forcing” the non-player characters to go to sleep.
v = The agent's current value for the specific need.

When all the needs have been calculated, they will be sorted by the biggest need.

$$\max [N1,N2,\ldots,Nn]$$

The top three needs will decide the Behavior tree-sequence that will be selected. When a behavior has been selected by the agent, for example, a lumberjack has decided upon the lumberjack-work sequence, described at, see Section 3.6.1, it’ll begin by finding the nearest tree. The equation to do so is the standard Euclidean distances equation. When all the distance has been calculated, they will be sorted by the shortest order first.

$$\min [d1,d2,\ldots,dn]$$

The object, which in this case is a tree, with the shortest distance to the agent will be selected to be cut by the lumber jack.

3.6 Villagers

The inhabitants will serve different essential roles;

1. Farmer: The farmer will plant and harvest resources necessary for the baker to bake food.
2. Lumberjack: The lumberjack will harvest wood for the bakery to function.
3. Baker: The baker will bake food for the village, a process which uses the resources acquired by the farmer and the lumberjack.
4. Traveler: The travelers will simulate people not from the village. These will mostly hang around the town square and the tavern.
5. Hero/Adventure: The Hero is the tester character. The hero is tired and weary from combat, seeking refuge or food in the village, before departing for new adventures! This character is **not** controlled by any AI.

The villager decision making is explained at Section 3.5.

The Behavior tree implementation followed the implementation described in Chapter 1.2. The trees are similar in many ways. All contain a sequence for socializing, eating, sleeping, wandering around and all but one contains a tree for working, however, the detailed implementations may differ slightly.

In general, the implementations follow these patterns:

1. Socializing is a sequence. It starts out by trying to find a villager to talk with, or finding the tavern. Once any of these has been successfully executed, the AI will proceed with walking towards a villager or tavern, and finally socialize.
2. Sleeping is a sequence. The AI will start out by localizing its home, then walk to it and sleep until rested. The traveler doesn’t have a home, and will instead proceed with sleeping in the tavern.
3. Wandering around is simply a way for the AI to go on a short walk-about in the village.

Working and eating differs in implementation. They will be described below together with their respective behavior tree.

3.6.1 Villager details

Please note: These sprites are the property of The Pokémon Company and Nintendo Co., Ltd.
The sprites were chosen due to their familiarity to the test users.

Baker

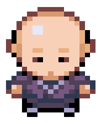


Figure 5 - The village baker.
See Appendix A

Baker behavior tree

1. Working: Working is a sequence. The baker will begin with gathering the resources necessary to bake bread. Both bread and wheat must be gathered from their respective storage facilities. After this, it will walk to a bakery. When walking to the bakery, it can either walk directly to the bakery, or take a detour. Once at the bakery, the baker will bake. When the baking is done, it will store the bread.
2. Eating: The baker will eat at the bakery. If he is not there, he will walk over to it and eat some bread.

Farmer

Figure 6 - The village farmer.
See Appendix B
Farmer behavior tree

1. Working: The farmer wants to plant, tend to, and harvest crops. The farmer will begin with walking to a farm tile. If the farm tile is empty, the farmer will plant a crop, and move on to the next one. If the tile is farmed, it will tend to the crops and water them. If the tile is watered and planted, it will harvest. It'll put the harvested crops into a storage for the baker.
2. The farmer will, depending on need, select one of several sequences for eating. The first one, the bakery sequence, will make the farmer head over to a bakery and buy some bread. The farmer will either consume the bread directly at the bakery, or bring it home. The second sequence, the "house" sequence, will see the farmer head over to one of the stands at the village square and buy some food and bring it home to eat. The third sequence, the "tavern" sequence, is more likely to trigger if the farmer wants to socialize as well. This sequence will see the farmer walk to the tavern, and simultaneously eat and socialize.

Lumberjack

Figure 7 - The village lumberjack.
See Appendix C
Lumberjack behavior tree

1. Working: The Lumberjacks work cycle is straightforward: Find a tree, walk to that tree, cut it down, and put the lumber in a storage facility.
2. Eating: The lumberjack follows the same eating sequence as the farmer.

Traveler

Figure 8 - The adventurer.
See Appendix D
Traveler behavior tree

The traveler is different from the other villagers, as it's not a part of the village. It doesn't work, or produce anything. It can generally be found around the tavern, and its relatively simple behavior tree visually displays this difference.

1. Eating: The travelers can only eat at a tavern.
2. Sleeping: The travelers can only sleep at a tavern.
3. Adventure: The traveler leaves the village to go to an adventure, and returns later when it is "done" to rest

3.7 Discussion

The following parts discuss the solution.

3.7.1 Choice of game engine and framework

The choice of engine comes down to previous development experiences and relative ease of use. Other engines such as Unity3D or the LibGDX framework for Java were also potential candidates, but the Monogame framework was chosen because of the previously mentioned experience.

Had we chosen Unity3D, we could easily have made our village in a 3D environment instead of 2D. The effects concerning the impressiveness of the overall experience for the test user are something we do not really know, as there are pros and cons to both 2D and 3D. Unity is an excellent game engine, however, our experience in Monogame is much greater. This is one of the reason why we decided on using Monogame instead. In addition, Monogame together with Tiled allowed us to create a village very quickly compared to how long time it would've take us to do it in Unity.

When it comes to development and visualizing, Unity has an upper hand, because Monogame is a framework with only a game loop and a graphics pipeline. As a developer in Monogame, you must build everything up from the ground up, nothing can be taken for granted, but the payoff is that it gives the developer more freedom and control over the software.

Overall, we believe that we made the right choice of engine and framework for our project.

3.7.2 Villager role discussion

Early during the production, we intended on having more villager roles, such as children. These children would simply walk around or follow their parents, but these were scrapped because of time constraints. It would be interesting to add on more roles, such as these children, smiths, miners, soldiers, all of which was discussed at some point, but the time cost of implementing and testing all of these would have been too great.

The current four villager roles were chosen due to their relationships. The baker needs two different raw materials, which can be refined for something more important, in this case, bread. These dependencies make the village more organic in our opinion. Adding smiths and miners would have added to this interesting dynamic, but at an expanded scope, both in terms of time and villager relationship.

Had we had more time, we would have not only added more villager roles, but also expanded the size of the village, perhaps making it more resemble smaller town with different districts. In the future, perhaps we could even have time to develop some sort of friendship-bonding mechanic, where villagers living close to each other greatly favoring each other's company rather than the company of some strangers from another part of the town. These mechanics would demand a lot of time to develop, not to mention testing.

3.7.3 Decision making discussing

Many small decisions concerning the planning can be discussed. For example, should the planner plan two steps ahead? Three steps? Four? Initially, we chose three steps ahead, but this led to situations where it was midnight, but the agent had still planned on doing work and eating, thus not being able to sleep until early morning. These issues were resolved by heavily prioritizing sleep during night, as well as forcing the agents to replant every two steps. We found that two steps ahead work for our day/night cycle, as it still allows them to stay up late, but eventually go to bed at a somewhat reasonable time.

3.7.4 Different villager techniques

AI systems in games have about 10-35% of the CPU power reserved to them (Schwab 2008), however, this thesis is not about performance issues, limits and cost. Due to a lack of time, we will not be able to compare this type of implementation to other types of AI systems, such as cycling scheduling (Zhao and Szafron 2014). It is however a topic for further research.

4. User study and tests cases

This chapter will explain in detail the progress of testing the AI in greater detail.

4.1 Testing process

The testing sessions followed this structure.

1. The test subjects were briefly interviewed about past gaming experiences. If they had played any RPG-games, it was noted for further post-test interviews.
2. The test subjects were introduced to the concept of the village, and their role in it, as well as how they can interact with the world. The test subjects were also allowed to freely traverse the world to learn the controls.
3. The test subject was introduced to their task, which, depending on the test, is either
    - 3.1. One test supervisor will control one (1) AI character with the test-controllers as specified in Section 3.13. The user will be given 4 minutes to spot one, after which they will be asked to explain who they think the test supervisor controlled. This will be repeated 2 times.
    - 3.2. This will be conducted in the same manner, but only one of the sessions contained an actual human supervisor controlled character, two of them only featured AI controlled agents. The reasoning behind this will be discussed later in Section 5.

Test subjects were chosen liberally from diverse groups of people. Test subjects are asked to fill in their previous gaming experiences, along with if they would comfortably call themselves a gamer (Appendix H). Important to note is that we do not intend on specifying what makes someone a "gamer". The testers own opinion is the only thing we care about here.

4.2 Questionnaires

Several questionnaires were made. They are similar, with some minor differences. The intro page (Appendix E) and Post-test page (Appendix H) is the same for all testers. The intro page will provide us with some basic information about the user. The Post-test page will give us some general feedback about the testers opinions and previous gaming experience, which is valuable information for us.

In addition to this, three more questionnaire pages was designed, with different intentions in mind. The first of these (Appendix F) is designed to be more open. For example, the first question is "Do you believe you could spot the human controlled characters". This means that the tester will provide us with what they *really* perceived. The second questionnaire (Appendix G) contains no references at all to any human controller. We will, however, mostly use the open questionnaire (Appendix F), as it allows the tester more freedom in answering. The reasoning behind this will be discussed in Section 4.3.

4.3 The test groups

Four different tests will be carried out. The major difference in these tests are what we say, and what we do.

Table 3 - The test case differences.

What we say	What we do
“We will control one of the villagers”	We control one of the villagers
“We will control one of the villagers”	We don’t control one of the villagers
“We will not control any of the villagers”	We control one of the villagers
“We will not control any of the villagers”	We don’t control one of the villagers
“We will maybe control one of the villagers”	We control one of the villagers
“We will maybe control one of the villagers”	We don’t control one of the villagers

This means that in half of the tests, the tester will be intentionally misinformed about our active participation in the test session.

A major part of this thesis is about the perceived realism by the user. In “*Believable Agents: Building Interactive Personalities, School of Computer”* (Loyall 1997), Bryan Loyall argues that the single most key factor for believable agents is personality.
The villager AI-system that we have proposed, use the “weighted need” where certain villagers have different priorities, creating an illusion of personality. To test this more in-depth than simply stating their task, we intend to use misinformation. By giving misinformation, yet still asking question about what they think they saw, we believe we can get more interesting feedback.

For example, what does it mean if a tester is told that there is a human controlled character, and believes to be sure about having spotted the test supervisor, yet we didn’t control anything, or vice versa.

Something to consider about misinformation, is that if the testers believe us too much, they might not spot even think about looking after the tester. If this happens, the feedback might not be too valuable.

The maybe question will be interesting, because this allows the tester to answer with most freedom. Since they are told someone might, as in we aren’t stating our interactions, control one of the villagers, they will more likely look harder and try to spot something, even if no one is controlling.

4.4 Questionnaire discussions

The following section will discuss the reasoning done in Section 4.

4.4.1 Open write-in questions versus scaled questions

There are pros and cons to both types of question formats, and in the end, we decided to use both. The Likert Scales are amazing for gathering bunks of easy-understandable data, which can be used in graphs and tables, but they hide a lot of the private opinions. By also using write-in questions, we hope to receive valuable private opinions, which, while harder to represent, can provide valuable insight as well.

5. Results

This section presents the feedback from the testers.

5.1 Testing cases

Before each test was performed, the tester was told different statements regarding whether the supervisor would also be playing the game, trying to blend in as an AI. The testers would be told that the test-supervisor would either, maybe or would not be controlling a character. But regardless of what was said, it did not have to be true. The following tables will show what was said during test the test cases, each tester performing two cases.

Table 4: How the testers were informed**.**

TEST CASE 1	Tester is informed that supervisor would be controlling	Tester is informed that supervisor could be controlling	Tester is informed that supervisor would not be controlling
Number of testers	3	12	5

Table 5: How many times the supervisor did control a character in the village

TEST CASE 1	Supervisor did control	Supervisor did not control
Number of testers	12	8

Table 6: How the testers were informed

TEST CASE 2	Tester is informed that supervisor would be controlling	Tester is informed that supervisor could be controlling	Tester is informed that supervisor would not be controlling
Number of testers	16	0	5

Table 7: How many times the supervisor actually did control a character in the village

TEST CASE 2	Supervisor did control	Supervisor did not control
Number of testers	14	7

5.2 Write-in question results

The questionnaire feature seven open-ended question to collect qualitative data. Three of them was asked after each test session, and the last four after both tests was done.

5.2.1 First form

How would you describe your past gaming experiences? How would you identify yourself?

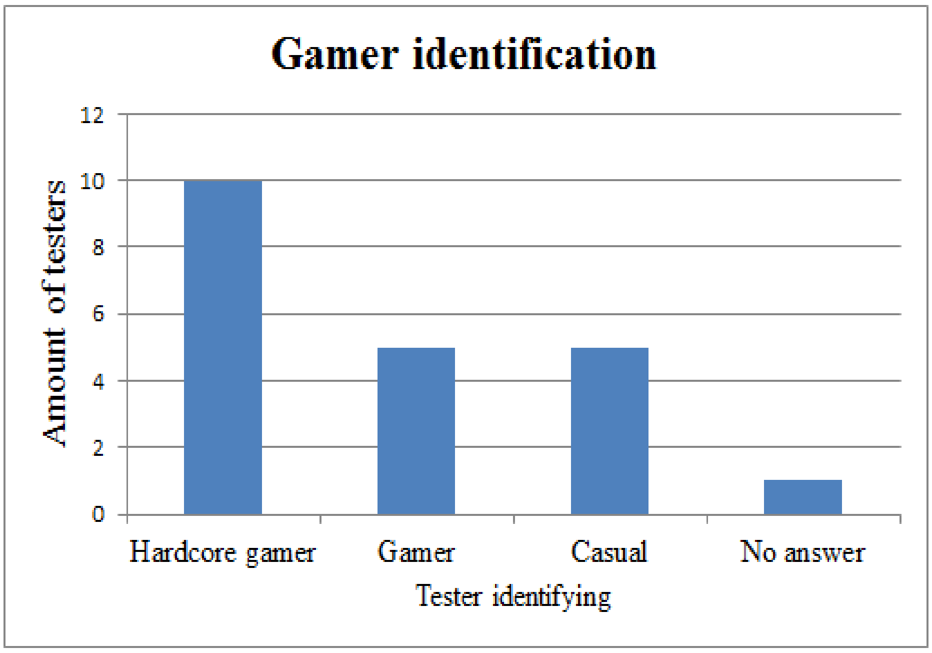


figure 9: What the different testers identified themselves as.

Do you think any of the villagers were directly controlled by a human player?

Table 8: Data showing what if testers believed the supervisor controlled a character during the tests

TEST CASE 1 AND 2	Testers that believed supervisor controlled a character	Testers that believed supervisor did not control a character
Number of testers	21	18

Who did the human player control?

The tester could write who they thought the supervisor controlled. In order to streamline this, each villager had a name above them, allowing the tester to more easily describe who they thought the supervisor controlled. By those who answered, a few managed to find the supervisor, as shown in figure 10 and in table 9-11, based on what they were told.

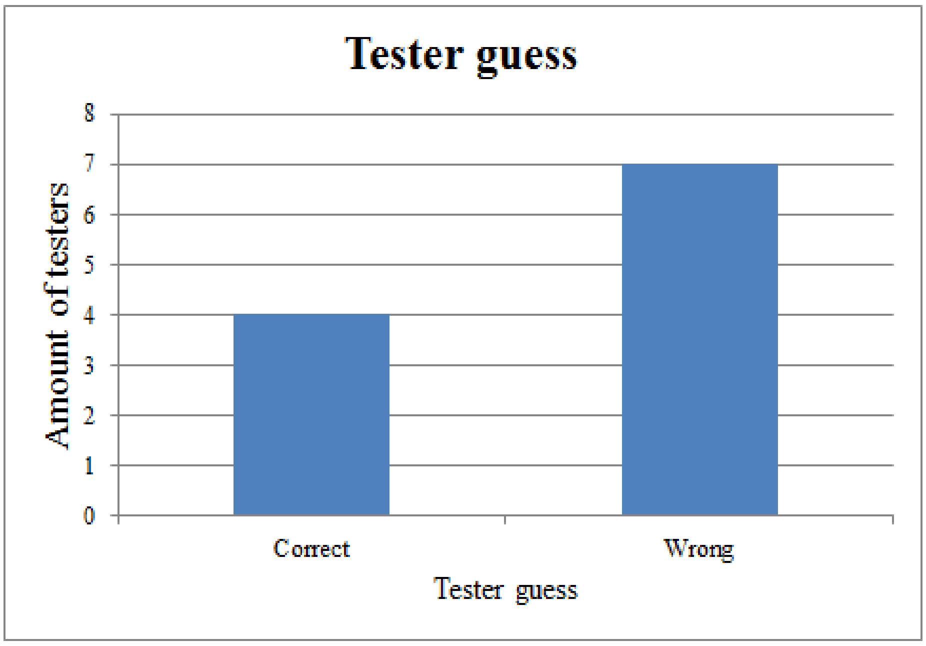


figure 10: Eleven testers guessed on who the supervisor was controlling.

Table 9: More detailed. Table showing amount of correct and incorrect guesses sorted by being told supervisor controlling a character. In total, 19 testers were told supervisor did control.

Testers told "supervisor controlled a character"	Correct guesses	Incorrect guesses
Number of testers	3	4

Table 10: More detailed. Table showing amount of correct and incorrect guesses sorted by being told supervisor did not control a character. In total, 5 testers were told supervisor did not control.

Testers told "supervisor did not control a character"	Correct guesses	Incorrect guesses
Number of testers	0	1

Table 11: More detailed. Table showing amount of correct and incorrect guesses sorted by being told supervisor maybe controlling a character. In total, 17 testers were told maybe.

Testers told “supervisor maybe controlled a character”	Correct guesses	Incorrect guesses
Number of testers	1	2

What, in your opinion, made the supervisor player not blend in?

Considering the answers gathered by this question, these were the most common ones that were pointed out by the tester.
In those tests were the supervisor was controlling a character in, few noticed some different with the characters movement that the supervisors was controlling. This quote summarizes that observation form the testers.
Swedish:
Original quote 1: *“Diagonal rörelse med ansiktet riktat "söderut" är inte vanligt bland AIn, men spelare gör det alltid.”*
English:
Translated version quote 1: *“Pointing it’s[AI’s] face south while moving diagonally is not common among the AI, but the player is always doing it”*

Among answers from the testers where the supervisor was not playing during the test, these quotes summarizes the responses:
Quote 2: *“I was mainly looking at how the characters got from point A to point B and they had certain ways for their behavior when they were walking on a road or what not. Some npcs had some weird walking patterns but I think one npc was the one who stood out the most just because he didn't walk in the middle of the road when walking from his house up to the market like the rest of the AI.”*

What should the supervisor do to blend in more?
The testers answered either that they don’t know, or that the supervisor was doing an adequate job.
Quote 1: *“I don’t know”*
Quote 2: *“just keep doing what you were doing.”*

5.2.2 Post-test form results

In this subsection, we will present the results from the post-test forms.

(If having previous gaming experience with any of these games) - How would you compare our villager AI to villager AI seen in games such as Pokémon, Final Fantasy, and the Witcher?
Much of the response from the tester were mostly positive to the AI system. But some did not feel the same. Here are some quotes that will give some overall response to how the testers felt by the AI system. These quotes provide us with some general feedback on what the testers felt.
Quote 1: "It looked a lot more believable than Pokémon and final fantasy, since the AI in those games usual follow strict patterns, whereas here, I was not able to discern any typical pattern the NPCs followed. The one that made it feel most like an AI was playing the characters, was that the movement looked quite "jumpy" especially where the turn angles were very sharp and looked like they were always made with the same angle. I would say that it was similar to the Witcher, except that there were no gatherings of people outside conversing. I only noticed this one time, and if that was occurring more frequently, I think it would look more lifelike"
Quote 2: "good, can't recall any fancy AI movement in Pokémon, I didn't see any of the villagers conversing, though I was told they could. To increase realism, maybe have them stand still? or move in a small area to simulate a more relaxing behavior, but it was great. Maybe have groups of more than two conversing"
Quote 3: "Most games have NPCs just standing around and following a set path so it was better in some ways. When NPCs strayed from the city paths a bit, it had me a bit fooled."
Quote 4: "The AI agents moved around seemingly fulfilling some sort of need. Although I did not feel there was much social interaction between different AI agents. I would compare this to Skyrim, where agents are not stationary."
Quote 5: "It doesn't feel "realistic" as you state it, however it feels very "levande"[alive] and that everyone has something to do, which I like in AN hero rpg."

Question: Do you believe, in general, that games would gain from implementing this type of AI in villagers? (Yes / No)

1. *"If yes, why?"*
   Quote 1:
   "It was not easy to identify patterns in their behavior, and the people would sometimes turn around and walk a unique way, like they forgot something. It made the world more vivid, and the people looked like they had goals, rather than just doing random actions. Like in most games NPCs are dedicated to one single action, but here they were moving around, doing all kinds of different things."
   Quote 2:
   "Makes the game world feel more immersive and it makes it more interesting for the player to interact with."

2. *If no, why?*
   Quote 1: “I'd rather answer "I don't know". I think it's hard to evaluate the AI in this kind of setting. A smaller environment with more distinctive characters would make it easier for me to "get to know" the characters so I could tell if their behavior made sense. Of course, I have no idea how the AI works in this case so I don't know if it's good or bad.”

5.3 Likert scale results

The questionnaire featured three Likert scale questions that the tester could answer. One of them ranged from “very bad (1)” to “very well (5)” the other one “not realistic (1)” to “very realistic (7)” and the last one “I disliked it (1)” to “I liked it a lot (7)”.
figure 12 shows how they felt about the village realism and how they liked it.

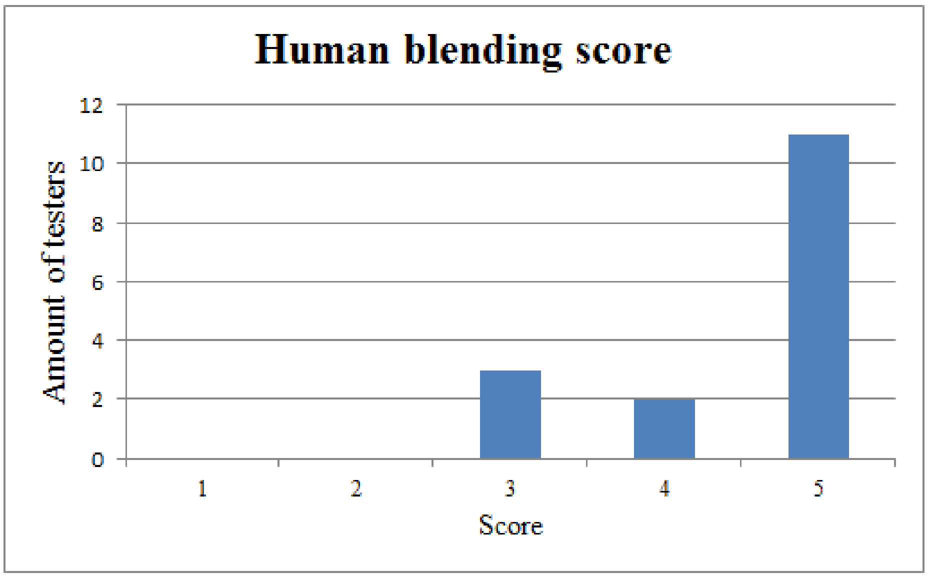


figure 11: Testers were asked to rate how they thought the supervisor managed to blend in among the AI characters in the village, from 1 to 5

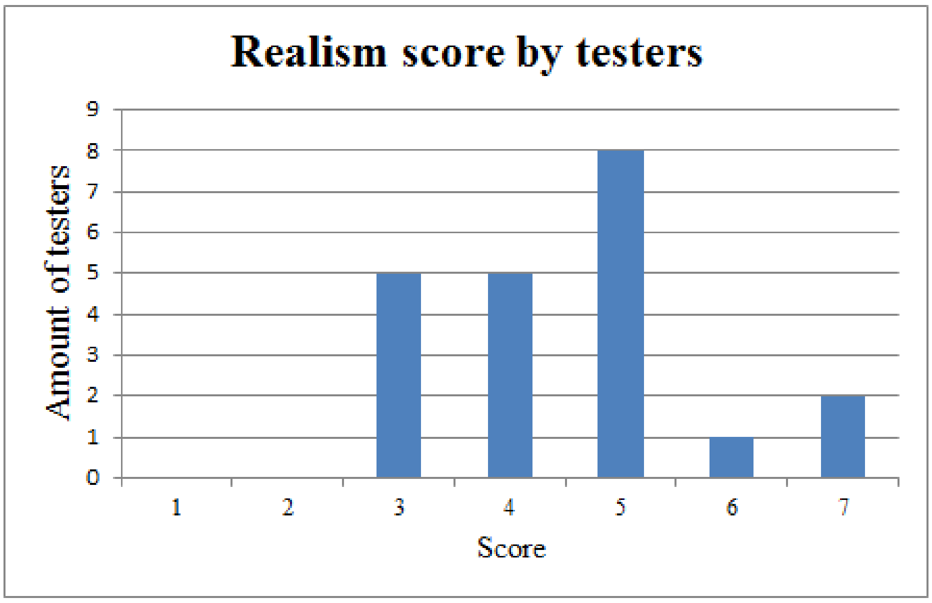


figure 12: Testers were asked to rate what how realistic they thought the AI behaved in the village

6. Analysis

This section of the thesis will analyses the results presented in Section 5.

6.1 Differences in tester's awareness of a supervisor

When being told "Yes, the test supervisor will control someone in the village", at no point did a clear majority of testers believe that they had manage to spot the human supervisor controlled character.

When being told no, there is no human supervisor player, a majority, 80%, of testers believed us. The reason behind this may be what was discussed in Section 4.3, whereas they simply believe that there is no human player, and don't attempt to notice anything different in the general villager behavior.

When being told maybe, there were some big deviations in the results. In the first set, 12 people were told maybe, and ten people believed that the supervisor did control a villager. In the second set of tests, only two out of five testers believed that the supervisor controlled a character. This is also interesting considering that out of all total *maybe* test cases, seventeen, only one tester managed to guess correctly.

6.2 Post-test questionnaire

We received information from the post-test questionnaire, see Section 5.2.2, the conclusion form, which contains general write in opinions and some Likert scale questions.

When asked: “How would you rate the “realism” of the village AI? Compared to other games, do you think this village felt realistic?”, the results were mixed, although leaning positive. Five people gave it a rating of 3/7, with general complaints such as thinking it was strange that many villagers were up late at night, or that they simply felt to random, that they could not see any pattern. Another five-people rated the game 4/7, a neutral, and the general feedback here is the AI Agents functioned in an overall effective way, but they lacked in certain specific ways. For example, some noted that the villagers felt “erratic” once they started moving towards a new location, and some noted that the agents didn’t seem to have any kind of meaningful interaction between each other.

The rest of the testers all answered positively. Eight people gave it a 5/7, generally stating that the villagers moved around realistically, and importantly, making it feel more life-like. They noted that compared to similar looking games, the AI didn’t simply stand around, they moved around, worked, went home during the evening and such. There were still negatives however. Some stated that they thought the pathfinding looked strange at times, and some of them also noted that they wished there would be villager interactions.

Among the last three testers, one gave it a 6/7 and two gave it a full 7/7. The most common feedback here is that they liked the dynamic feel of the AI, which gave it a more lifelike experience. Some even noted that they had hoped that the they themselves could have interacted with the villagers. All in all, 23.8% of the testers left a negative review of the realism, another 23.8% left a neutral/unsure mark, and the last 52.4% left a positive response. These responses are visualized in figure 12, Section 5.3.

When everyone was asked whether they thought games of similar nature to the ones mentioned in the introduction (*Pokémon, Final Fantasy, Dragon Quest, etc.*) would gain or lose, both in general gameplay and in realism, by implementing a similar form of villager AI, see Section 5.2.2, an overwhelming majority answered that they would indeed think they would. Many answered that they thought the dynamic feel would not only add replay-value, but also made the AI feel more alive. Testers generally noted that they thought a living moving village like the ones they tested would make these games feel more interesting. Some even noted that “Purely out of the activity from the ambient characters, it's great”, and that it “Makes the village feel more alive when AI agents are moving around”.

Three people answered that they thought games would not gain from using this kind of AI system. They generally wrote that they felt a bit overwhelmed and a bit confused. One noted that they, due to the lack of in-depth interaction options with the villagers, felt like they couldn’t get a feeling about what the different villagers were doing.

Another one noted that they thought the village should have been smaller, with a smaller number of villagers, so they could get a better feeling of the dynamics of the village.

6.3 Differences between gamer identifications

In the expected results, Section 1.5, we hypothesized that non-gamers would find the living village interesting, while we thought the core gaming crowd might find it a bit distracting. However, interestingly, nothing in the gathered results backs this claim. There are no clear patterns between hardcore gamers, core gamers or casual player.

Interestingly, among the three people who did not think other games would gain from implementing this type of AI, one identified as a hardcore gamer, one as casual, and one left the field blank, making it impossible to draw a real conclusion,

Something that is worth noting, is that only players who identified as hardcore or core gamers managed to correctly guess if a test supervisor was involved or not, no casual gamers managed to guess right.

7. Discussion

This section will discuss the results and analysis from Section 5 and 6.

7.1 Quality of implementation

One thing that we noted, and that is reflected in the results, is that even small errors done by the AI will ruin the immersion. For example, as mentioned in Section 6.2, one small strange movement made by an agent was enough for the tester to identify the AI, and calling its movement "erratic". Another tester noted that a strange number of villagers was outside walking around in the village at 2-3 AM. This is something that might make the AI seem more strange and random than realistic to the human testing player, however, small amount of random behavior might also make the village seem more alive. Instead of every light disappearing at 9 PM, there'll still be some life, and as such, there might be more potential actions and events for the player, instead of simply heading for a tavern. Things like these must be corrected and thoroughly tested, or there is a chance that it might be counterproductive to the end goal of creating a more realistic AI behavior. However, if enough testing can be executed, a golden middle path can perhaps be achieved.

7.2 Time consumption

Crafting an AI based on needs creates the need to also program the environment around the AI, depending on the amount of realism that is desired. In our case, that meant we also had to design a day and night cycle, as well as different objects for the AI agents to interact with. These objects depend on the scope of the city, and they must be planned early on. In our little village, we did just fine with cut-table trees, storages, farms, and such. However, if you want an even more realistic village, perhaps even a city, the number of roles grow. For example, perhaps the growing city demands miners, smiths, doctors and other more specialized professions. And as the city dynamics grow, so too might the details of the communication between the AI agents. For example, should the city have a police force? If so, will it need the actual AI behavior that necessitates a police force? Will it feature laws and law-breaking? Will the city have AIs that are thieves and burglars since creation, or will dynamic variable force citizens to turn to criminality. In our opinion, an AI system of this depth might be amazingly interesting, but the time necessary to code such a system would not have justified such detail in our scenario.

7.3 Test group size

While our test group wasn't very large, Tullis and Bill argues that, while describing common myths about test sampling, states that a large test-group, while it might help to increase the level of confidence, is not necessary, and that it is fully possible to have a satisfying test group with only 20-30 testers. We do, however, believe that larger test groups should be considered for future AI research, due to the considerable number of testers leaving a neutral result. Everyone we tested was also within the same age range, 20-30 years old, and it could potentially be interesting to see how younger and older groups will react to the AI. That said, larger groups might be counter-productive, as it could lead to test results converging to an average mean.

7.4 Villager movements versus behavior

We never specifically mentioned that the testers were supposed to spectating the villager behavior in the forms, although, the testers were told this before each session. This could potentially contaminate the results slightly, as the testers only focused on how the villagers got around in the village, instead of how they acted and behaved. While we do not think this invalidates the results, we want to mention it, since one tester claimed this is how he or she spotted the test supervisor in the village.

8. Conclusion

Our first research question: "To what degree can the behavior system be implemented in a village-setting in a role-playing game where you do not typically see them?" To answer this question, we believe that the system is possible to be implemented in this type of environment, however, there is an issue concerning the time it takes to program and implement this kind of solution. This is discussed in detail in Section 7.2.

To answer the second question, “To what degree does this affect the perceived realism by players when compared to the classic approaches”, as seen in Section 6.2 and figure 12, 52.4% thought the realism was better compared to other games of similar nature, while 23.8% left a negative response, while another 23.8% didn’t want to submit a definitive response. That said, an overwhelming amount of people thought that the village felt more alive than games with a comparable setting, even people who gave it a negative review agreed on this, and thus, in this regard, we believe that this kind of AI can positively affect the realism of games.

8.1 Future studies

There is plenty of additional tests that can be performed in this area. One example, would be to compare the Behavior tree planning system with Cyclic Scheduling. This paper will not attempt to discuss or explain Cyclic Scheduling as specified in the Section 1.6, but the system has been successfully implemented in Bethesdas *Elder* Scrolls series (Bethesda Game Studios 2002) and *Fallout* series (Bethesda Game Studios 2004). How would players perceive the realism of these two systems compared to each other.

Another example and area of further improvement could be pathfinding. In our system, there are three levels of tile “cost”. For example, walking on a road cost 1, while walking outside the roads costs more. This way, the AI priorities walking alongside the roads. However, an interesting idea that we had, would be to see what would happen if we allowed a player to “pretend” to be a farmer. Every day, the player would have to manage working, getting food, interacting with other villagers, etc. While the player is doing this, some sort of heatmap data would be collected. Once enough data is gathered, it could potentially be interesting to see how this data compares with how the AI would plan and do everything. If the data is very different, it could potentially be used to re-define how the AI would plan and perform its tasks. Perhaps the data gathered by the human movement and interactions could be used together with a machine learning algorithm, and improve the realism of the AI.

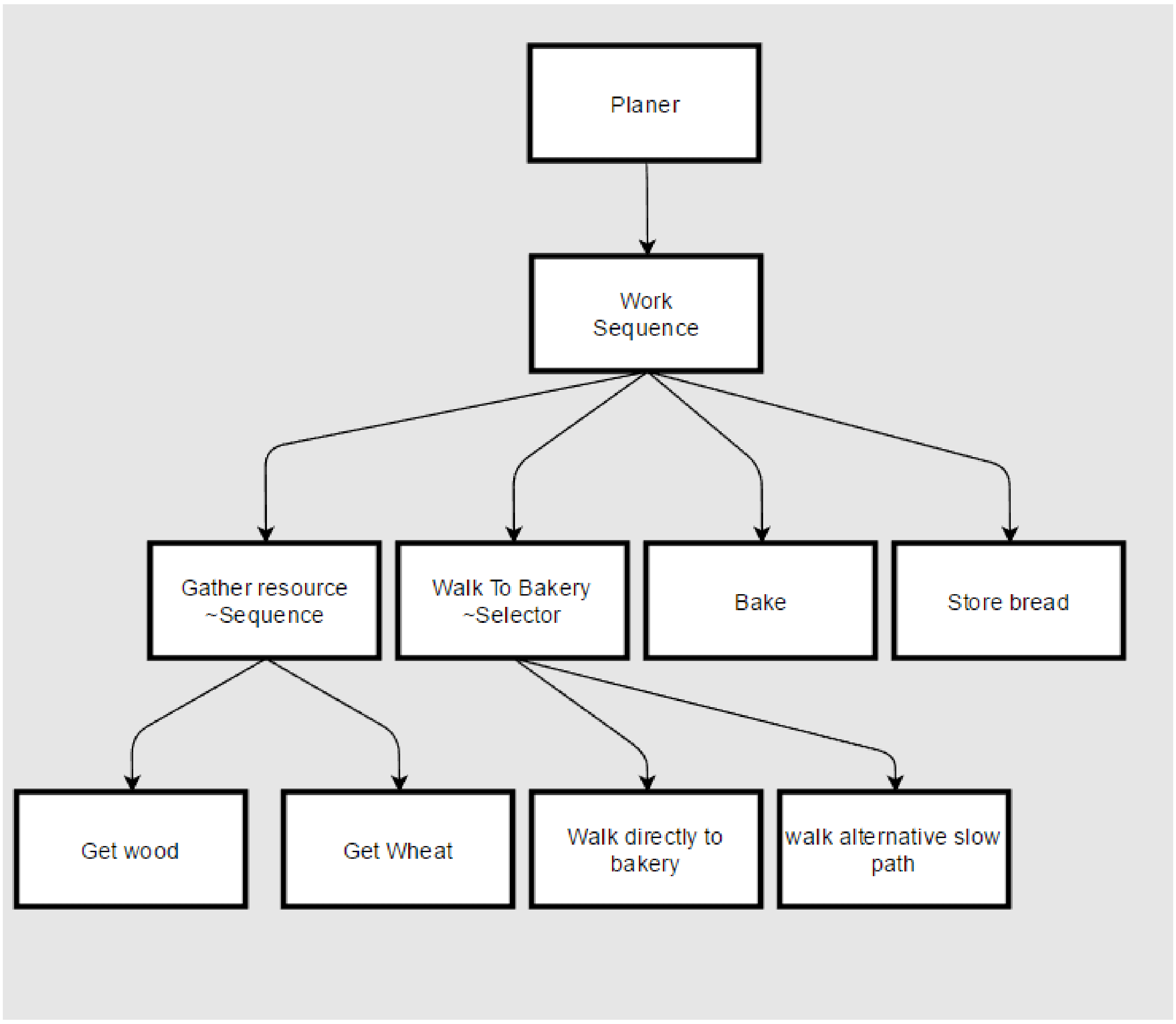
9. References

1. Maxis- Electronic Arts 2000-present, *The Sims* (series), video game, Electronic Arts, United States of America.
2. Game Freak-The Pokémon Company 2016, *Pokémon Sun and Moon*, Nintendo, Japan.
3. Game Freak-The Pokémon Company 1996-present, *Pokémon* (series), Nintendo, Japan.
4. Square Enix 2016, *Final Fantasy XV*, Square Enix, Japan.
5. Bungie 2004, *Halo 2*, Microsoft Game Studios, United States of America.
6. Lionhead 2003-2005, *Black & White* (series), Electronics Arts, United States of America.
7. Square Enix 1986-present, *Dragon Quest/Warrior* (series), Square Enix, Japan.
8. Square Enix 1987-present, *Final Fantasy* (series), Square Enix, Japan.
9. Ape/HAL Laboratory 1989-2006, *EarthBound* (series), Nintendo, Japan.
10. Bethesda Game Studios 2004-present, *Fallout* (series), Bethesda Softworks, United States of America.
11. Bethesda Game Studios 2002-present, *The Elder Scrolls* (series), Bethesda Softworks, United States of America.
12. Loyall, Aaron Bryan. 1997. *Believable Agents: Building Interactive Personalities. Diss.,* Pittsburgh: Carnegie Mellon University.
13. Sahlin, Jesper. Olsson, Victor. 2015. *A Smart Terrain based model for generating behavioral patterns*. Malmö University, Teknik och Samhälle.
14. Schwab, Brian. 2008. *AI Game Engine Programming*. 2. Edition, Boston: Cengage Learning.
15. Millington, Ian. Funge, John. 2008. *Artificial Intelligence for games*. 2. Edition. Burlington: Morgan Kaufmann publishers.
16. 2009. About Monogame. *Monogame*. [Website]. http://www.monogame.net/about/ (Accessed: 06-Mar-2017).
17. Getting Started with XNA Game Studio Development. *XNA*. [Website]. https://msdn.microsoft.com/en-us/library/bb203894.aspx [Accessed: 07-Apr-2017].
18. About Tiled. *Tiled*. [Website]. http://doc.mapeditor.org/manual/introduction/ [Accessed: 07-Apr-2017]
19. Tullis, Tom. Albert, Bill. 2008. *Measuring the user experience*. Burlington: Morgan Kaufmann publishers.
20. Simpson, Chris. 2014. Behavior Trees for AI: How they work. *Gamasutra*. [Blog]. http://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php [Accessed: 19-Apr-2017].
21. Isla, Damian. 2005. GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI. Gamasutra. [Blog]. http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php [Accessed: 17-May-2017].
22. Offermann, Philipp. Levina, Olga. Schönherr, Marten. Bub, Udo. 2009. *Outline of a Design Science Research Process.*

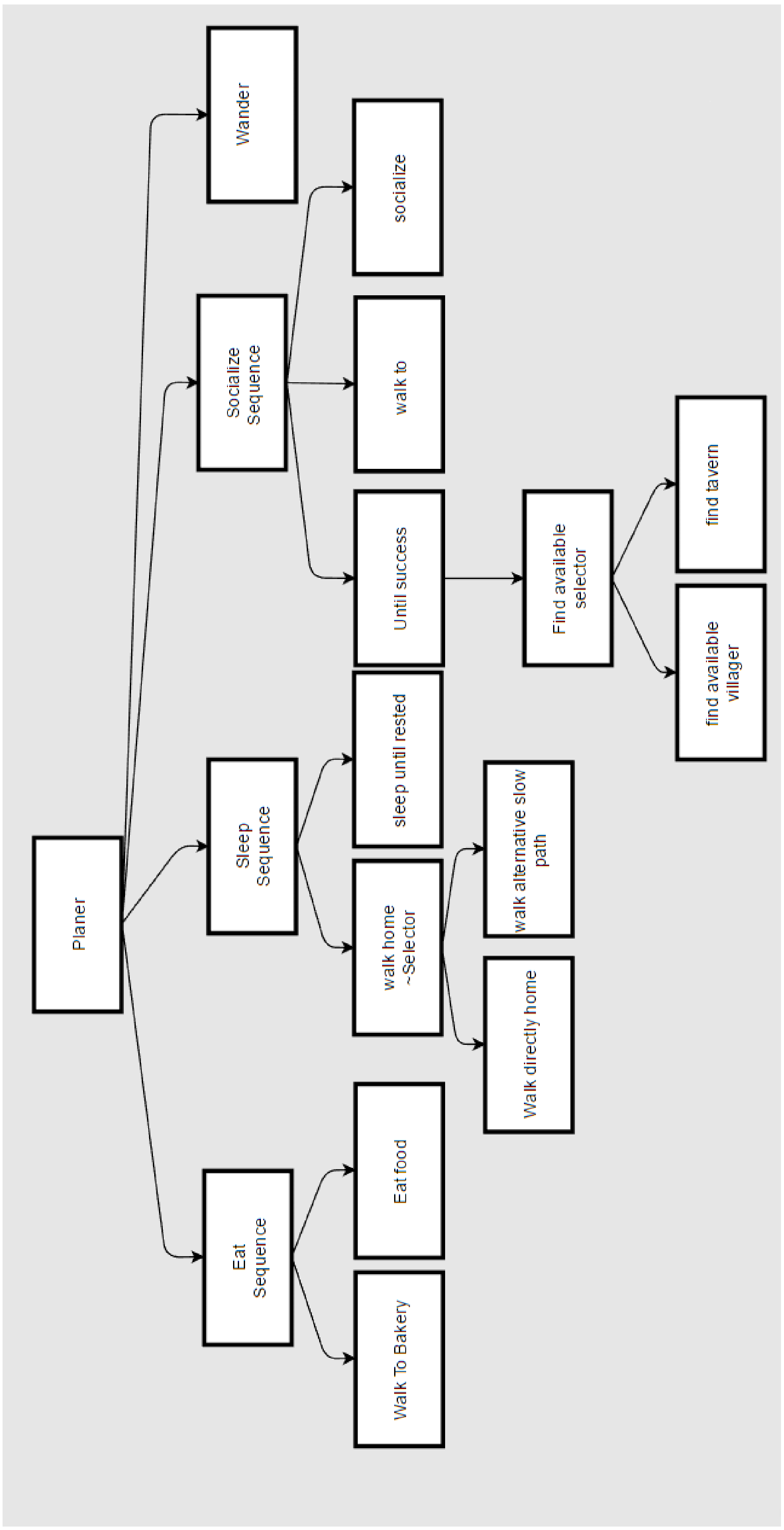
23. Yildirim, Sule. Stene, Sindre Berg. 2008. *A Survey on the Need and Use of AI in Game Agents.* Hedmark University College, Computer Science Department
24. Diller, David E. Ferguson, William. Leung, Alice M. Benyo, Brett. Foley, Dennis. *Behavior Modeling in Commercial Games,* Cambrige: BBN Technologies.
25. Zhao, Richard. Szafron, Duane. 2014. In Proceedings of the Tenth Annual AAAI Conference on Artificial
26. Intelligence and Interactive Digital Entertainment. *Using Cyclic Scheduling to Generate Believable Behavior in Games*. 2014. Raleigh, North Carolina, US.
27. Peffers, Ken. Tuunanen Tuure. Rothenberger, Marcus A. Chatterjee, Samir. 2007. A Design Science Research Methodology for Information Systems Research. Vladimir Zwass. *Journal of Management Information Systems.* United Kingdom: Taylor & Francis, Volume 24 Issue 3, Winter 2007-8, pp. 45-78.
28. Yannakakis, Georgios N. Togelius, Julian. 2015. *A Panorama of Artificial and Computational Intelligence in Games.*

9. Appendices

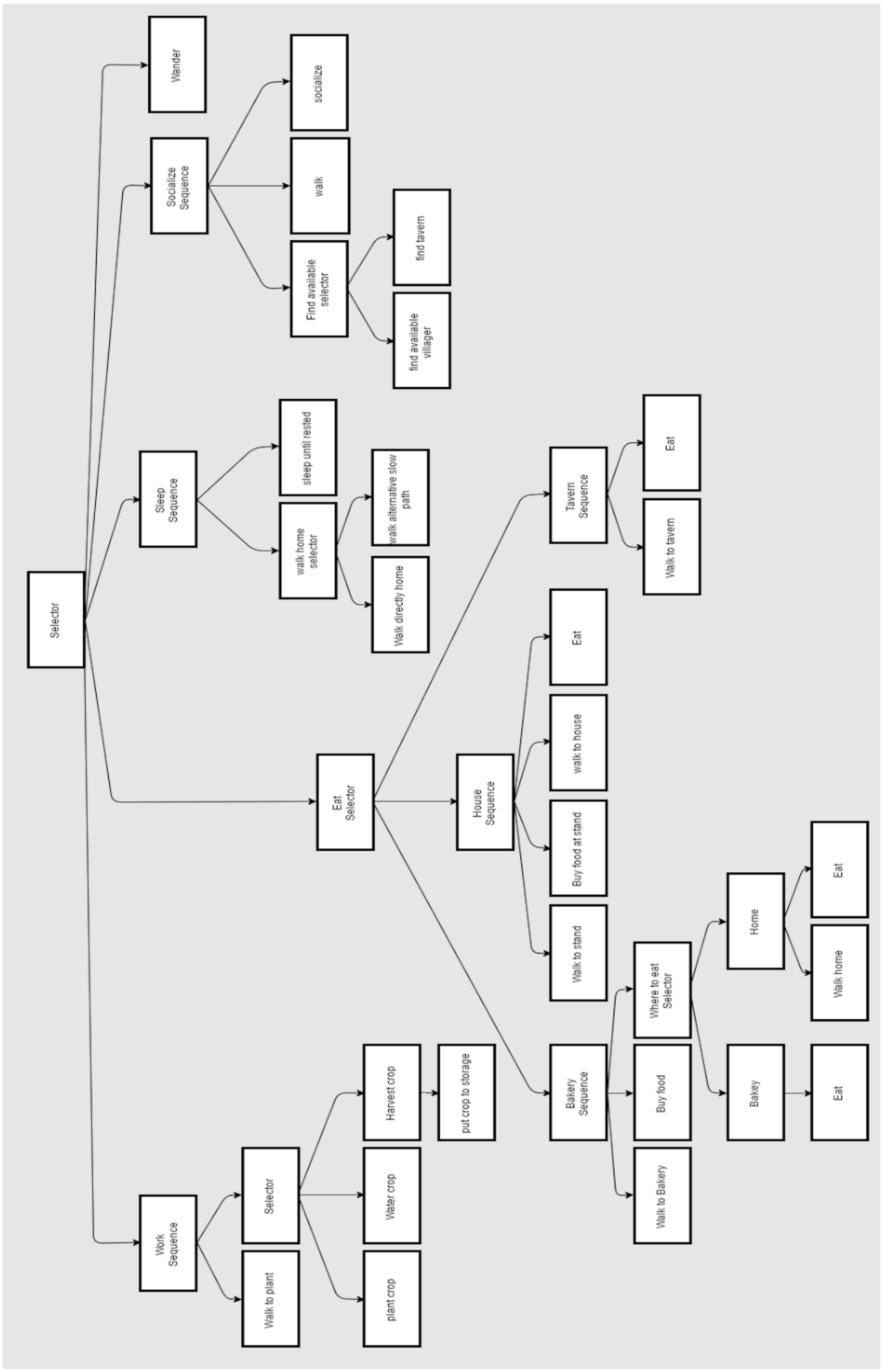
Appendix A (Baker - work tree)



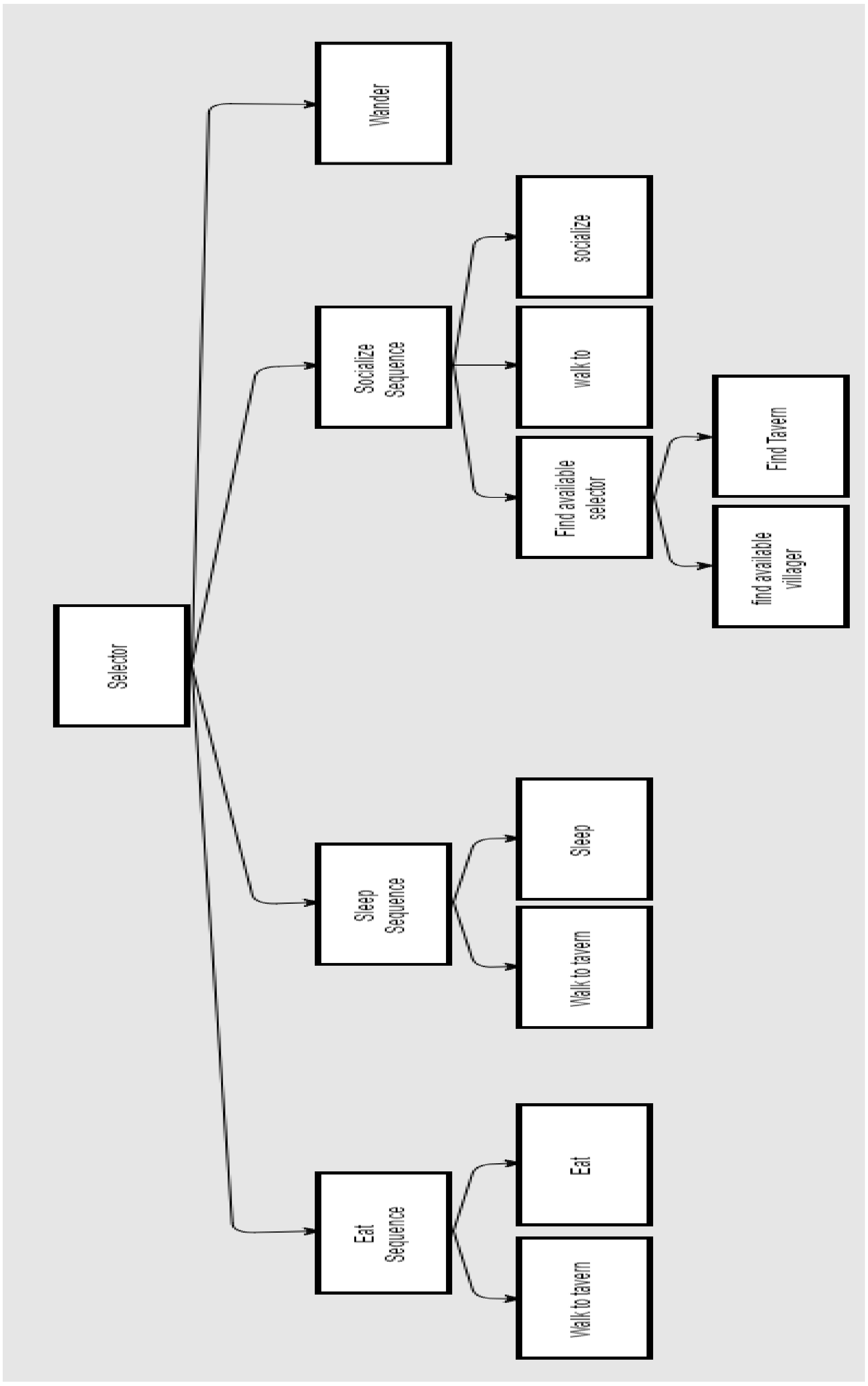
(Baker - The rest of the tree)



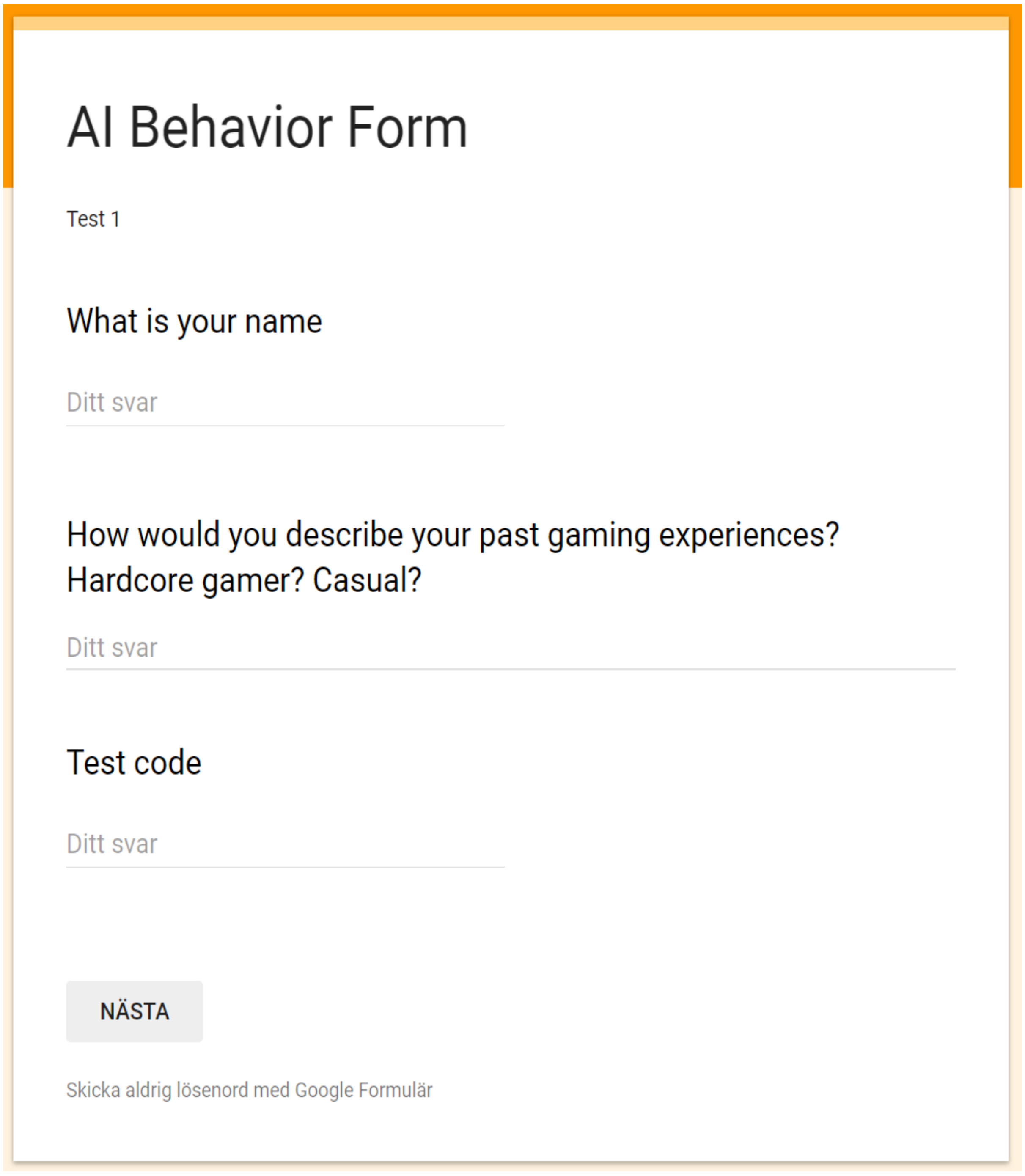
Appendix B (Farmer)



Appendix D (Traveler)



Appendix E (Intro Questionnaire)



AI Behavior Form

Test 1

What is your name

Ditt svar

How would you describe your past gaming experiences? Hardcore gamer? Casual?

Ditt svar

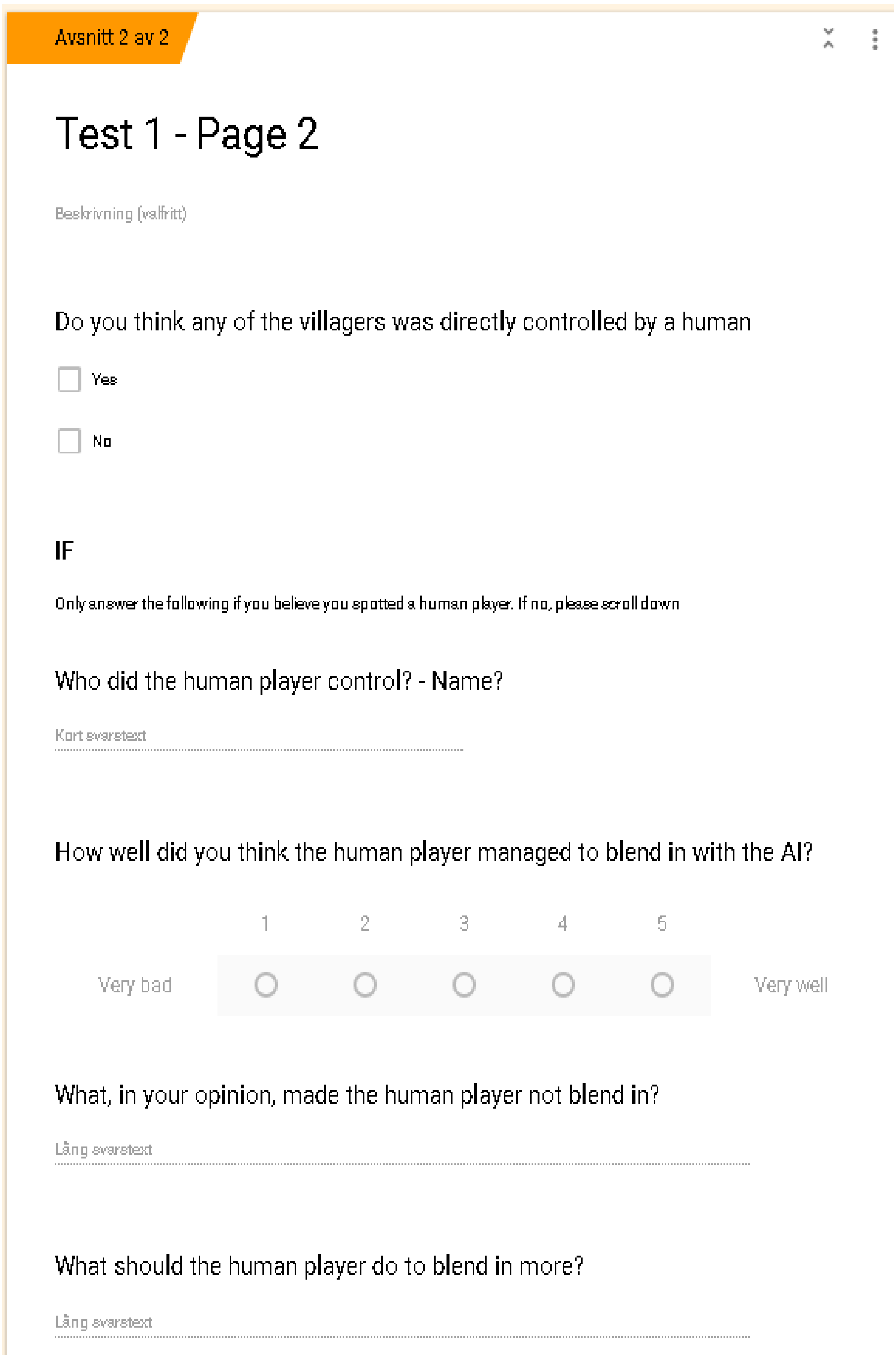
Test code

Ditt svar

NÄSTA

Skicka aldrig lösenord med Google Formulär

Appendix F (Questionnaire 1)



Avsnitt 2 av 2

Test 1 - Page 2

Beskrivning (valfritt)

Do you think any of the villagers was directly controlled by a human

☐ Yes

☐ No

IF

Only answer the following if you believe you spotted a human player. If no, please scroll down

Who did the human player control? - Name?

Kort svarstext

How well did you think the human player managed to blend in with the AI?

	1	2	3	4	5	
Very bad	○	○	○	○	○	Very well

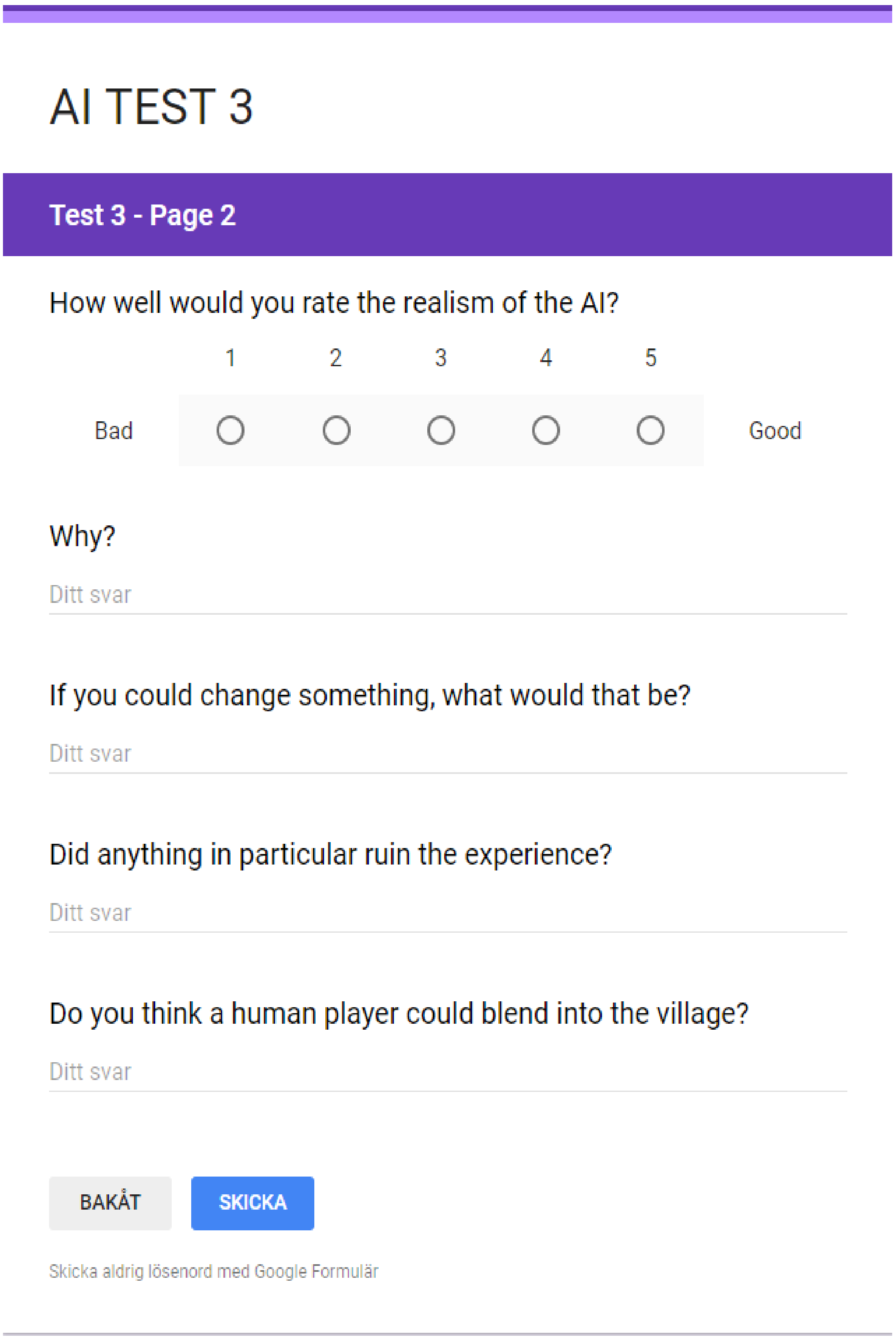
What, in your opinion, made the human player not blend in?

Lång svarstext

What should the human player do to blend in more?

Lång svarstext

Appendix G (Questionnaire 3)



AI TEST 3

Test 3 - Page 2

How well would you rate the realism of the AI?

	1	2	3	4	5	
Bad	○	○	○	○	○	Good

Why?

Ditt svar

If you could change something, what would that be?

Ditt svar

Did anything in particular ruin the experience?

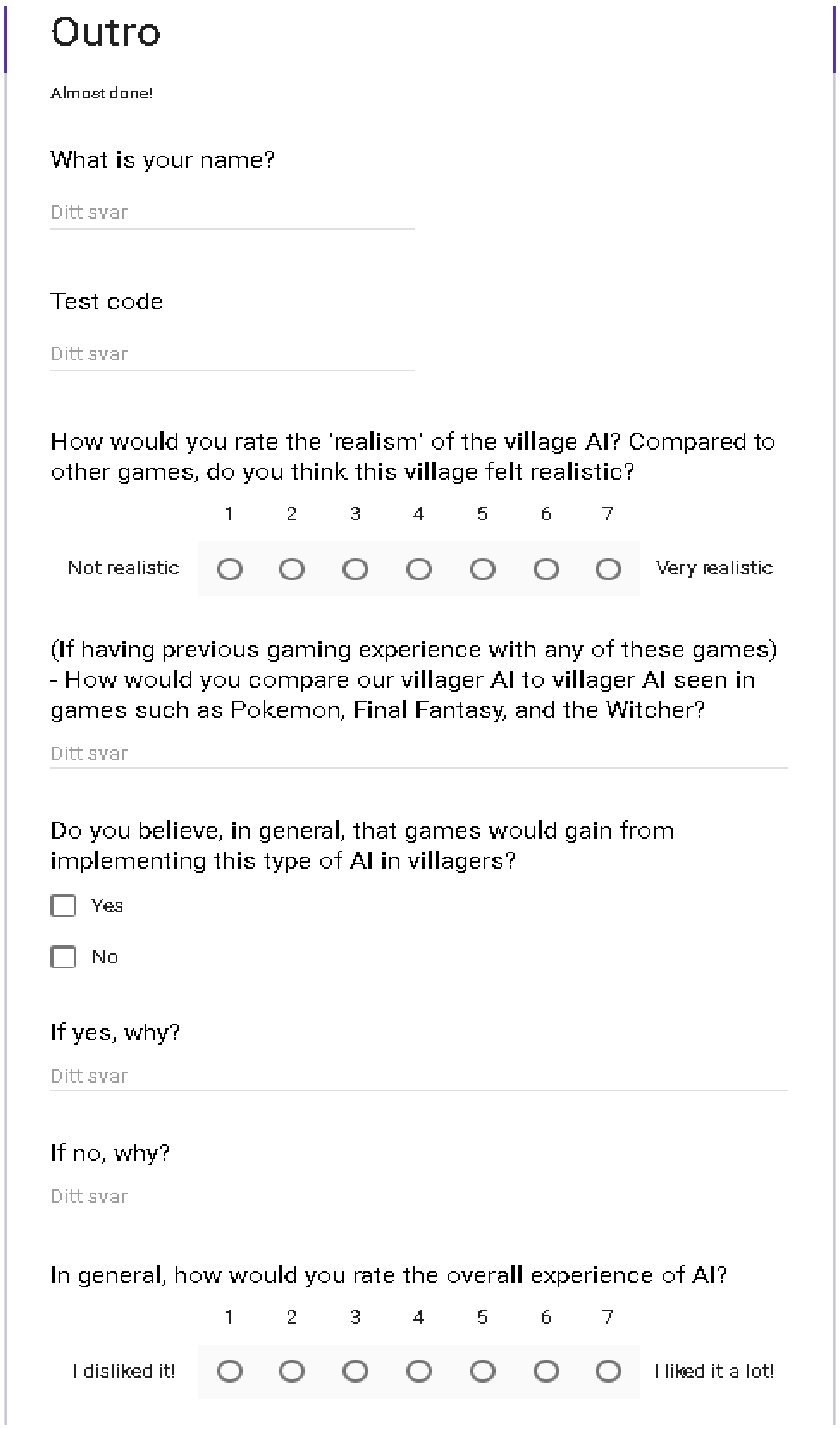
Ditt svar

Do you think a human player could blend into the village?

Ditt svar

Skicka aldrig lösenord med Google Formulär

Appendix H (Questionnaire last page)



Outro

Almost done!

What is your name?

Ditt svar

Test code

Ditt svar

How would you rate the 'realism' of the village AI? Compared to other games, do you think this village felt realistic?

	1	2	3	4	5	6	7	
Not realistic	○	○	○	○	○	○	○	Very realistic

(If having previous gaming experience with any of these games) - How would you compare our villager AI to villager AI seen in games such as Pokemon, Final Fantasy, and the Witcher?

Ditt svar

Do you believe, in general, that games would gain from implementing this type of AI in villagers?

☐ Yes

☐ No

If yes, why?

Ditt svar

If no, why?

Ditt svar

In general, how would you rate the overall experience of AI?

	1	2	3	4	5	6	7	
I disliked it!	○	○	○	○	○	○	○	I liked it a lot!