Believable Communication Between Non-Player Characters

Pedro Roquette Quadros Saldanha
pedro.saldanha@tecnico.ulisboa.pt

Instituto Superior Técnico, Porto Salvo, Portugal

October 2018

Abstract
Motivation for this work comes from the belief that there is some untapped potential is giving non-player characters (NPCs) in video games the ability to share knowledge between themselves. This dissertation proposes a model for how NPCs in video games can communicate between themselves in a more believable way. This model gives NPCs the ability to perceive events from their surrounding environment, allowing them to change their behavior based on those events.

We believe that by controlling how the information generated in the game world is shared between NPCs, we can change the way NPCs interact with the player. For example, in Middle-earth: Shadow of Mordor, the ability to give the player quests he needs in order to advance in the game changes the way the player interacts with NPCs. In this multi-agent system of NPCs, we use use of decentralized systems to give each individual NPC the ability to make his decisions independently from the other NPCs and from this decentralized behavior emergent gameplay situations can happen. We detail here the different techniques game designers can use to make a system like this one work, the difficulties they might expect and the possibility to increase the complexity of NPCs for future works.

Keywords: NPC communication, message propagation, emergent behavior, video game, believable communication

Introduction

Knowledge about the state of the world and what the player is doing on that world are very important requirements for any NPC to function properly. By enabling NPCs to share knowledge between themselves about the state of the world, over time they will change their perception about the world around them. One of the ways in which the behavior of NPCs can change is in role playing games (RPGs).

In RPGs, for example, NPCs that are able to give quests to the player require some knowledge of the world or the player in order to know if they should give the player the quest they hold. Traditional role playing games (RPGs) have always used some sort of quest based system to tell their story. As the player explores the world, he interacts with different NPCs with the ability to give the player the quests he needs in order to advance in the game.

Some games try to make “smarter” NPCs by improving their behavior and the way they interact with the world and the player. Some games like Middle-earth: Shadow of Mordor have NPCs that are able to make decisions independently from the other NPCs and act according to their own intelligence.

In a simulated world environment we can change and adapt rules as we please that would not be possible to apply on a real world situation. What makes the game “believable” is the ability of the player to recognize something in the game that is present in reality. The less believable a game is, the more a player is forced to learn the world rules, because when he perform an action he does not know what the outcome might be.

The fact that information generated by the world is globally available to all NPCs that inhabit that world constitutes a problem in a world where a more believable communication is desirable.

The solution we are proposing is a way for NPCs to share information generated by the player (or the world) between themselves, and as such being able to react and adapt over time to the changes happening in the world.

Background

Quest Systems

Quests in role-playing games can be described as tasks the player has to complete (alone or in a group) in order to gain some type of reward. When trying to complete a quest, the player interacts with the games world to unravel the quest or understand the plot behind the quest.

A quest is a “hunt for a specific outcome”, in contrast to simply winning a game. Quests could be understood as tools used in role-playing games to avoid putting players in a position where they only perform a repetitive action (also known as grinding).

Quests can follow a path through the story plot, and quest dependency graphs are directed graphs that represent not only the quests to be given to the player, but the prerequisites for that quest to be given and the order by which they are given to the player.

1. Smart Terrain

The game “The Sims” introduced an architecture known as “smart objects” to the game world. The main thrust was to offload a number of functions from the agent onto the objects in the environment. Each inanimate object in the game (that the Sim can interact with) contains two important categories of information: what benefit the object can provide for the Sim and how the Sim interacts with the object.

The behavior used in Smart Terrain is a good way of making NPCs in the world behave in a more realistic way. A social network that is based on random change encounters between NPCs has a better chance of spreading the correct information to the right NPCs.

The communication between NPCs can lead them to change their goals, and the Smart Terrain technique used on The Sims could be used to change the movement of the NPCs to different locations they are more interested in, playing an important role in deciding if NPC A crosses the path of NPC B, allowing both of them to start a conversation.

Emergent Gameplay
Emergent gameplay refers to complex situations in video games, board games, or tabletop role-playing games that emerge from the interaction of relatively simple game mechanics.\(^2\) Emergent gameplay can be classified as intentional or unintentional. Unintentional emergent occurs when creative uses of the video game were not intended by the game designers and can happen as a result of glitches, when the player finds other ways to play the game besides the one the designer intended for the player. Intentional emergent gameplay can happen as a result of a game mechanic like the one proposed in this model, where the communication between NPCs can lead to unexpected changes in the behavior of the world and the NPCs living on that world.

The main purpose of our model is to create an emergent gameplay where NPCs would behave in ways not expected by the designer. Although having a game with emergent gameplay is not always a feature a game designer would want to implement, in some cases, it can help a game have more replay value and captivate the player in ways that would not be possible without this type of gameplay.

Decentralization
Decentralization is the process of redistributing or dispersing functions, powers, people, or things away from a central location or authority.\(^3\) In a decentralized system, lower level components operate on local information to accomplish global goals. The global pattern of behavior is an emergent property of dynamical mechanisms that act upon local components, such as indirect communication, rather than the result of a central ordering influence.

The game itself is not in control of the data produced by the player while he is playing and has limited control over each NPC, who works as an independent entity, and his behavior is only influenced by his internal state and not the authority of the game. This can create very interesting situations and gameplay experiences during the game, since not even the designer knows what is going to happen.

Social Networks
A network is a system whose elements are somehow connected [4]. The elements of a system are represented as nodes and the connections among interacting elements are known as ties or links. Each node is then capable of sharing resources with the nodes it is connected to through his links.

A social network is a social structure made up of a set of social actors (such as individuals or organizations), connection ties, and other social interactions between actors. In an agent-based model, the agents update their internal state through an interaction with their neighbors and the emergent macroscopic behavior of the system is the result of a large number of these interactions.\(^1\)

Social networks can sometimes be organized in groups of agents who share similar properties. This networks form clusters of individuals, where each individual belongs to one or more cluster. There are two types of clustering. In hard clustering, an item is assigned to just one cluster, while in soft clustering each item can belong to multiple clusters with different strengths. In our model, we simulate soft clustering by giving different weights to NPCs different types of interests. The type of conversations they have will change as a result of having different interests.

In some social networks, not all ties have the same capacity. In fact, ties are often associated with weights that differentiate them in terms of their strength, intensity, or capacity [2]. We call this type of network a weighted network.

Approach
Our goal is to implement a system that allows NPCs to communicate with one another in a way that makes sense and mimic the behavior that happens in real life conversations. With this we hope to increase variability of content.

Information is shared inside an environment between non-player characters, making them more realistic and relevant to the player as he interacts with the environment over time. It is our hope that the player can see, interact, and exploit the communication happening between NPCs to his advantage, and that he can understand the similarities between how NPCs communicate and how people talk to each other.

The game world consists of different types of objects with different types of behavior:

1. NPCs that share information with each other at any time.
2. Entities that are able to perceive events happening on the world and create some sort of information about that event.
3. Player who acts upon the world, creating some sort of information over time.

NPC Communication Model
Let’s first consider all the steps that happen when NPCs receive and share information between themselves and that must be present in the final description of this model:

1. There is an event happening in the world. This event can be caused by the player over an entity, by that entity itself or even by the actions of NPC over NPC.
2. An entity creates a message when an event happens to that entity. An entity must have mapped all the possible messages it can generate as a result of different actions done to it.
3. The message generated is spread to all NPCs close enough to the entity to see it or hear it. Only sending messages to NPCs that are in perception distance can be an option here.
4. NPCs evaluate the message received and decide if they want to keep it or discard it. Since there is no reason not to keep a message if they have memory space available, the message received will only be evaluated if there is no more space.
5. NPCs must have a desire to send and receive messages. An assertiveness value measures how much they want to send a message and a cooperativeness value measures how much they want to receive a message.
6. When two NPCs cross paths, there are a few things that must happen in order for them to start a conversation:
   (a) They first see if they are willing to talk and the other NPC to listen (or the other way around)
They then check if they have any information to talk about between themselves that is of any interest to both of them.

If there is nothing for them to talk about they ignore each other and continue on their path.

If there is some message to talk about and all conditions for them to talk are satisfied, they stop their current behavior and start talking with each other.

The message received starts increasing in decay over time, becoming less relevant. At this point two different things can happen now to that message:

The message decays so much that it’s lost from memory. At this point the message is removed to allow for new ones to get inserted without the need to evaluate them.

If **NPCs** talk about that message again, that message is reinforced and the decay resets to the same value as a new message.

The cycle repeats itself, as new event happen in the world again.

This model can be subdivided into two distinct activity diagrams. On **Figure 1** we can see how a message is generated and reaches an NPC from the moment the event happens and finally reaches the player.

On **Figure 2** we see how the NPC behaves, how he interacts with other NPCs and what does it take for two NPCs to start talking. We can also see here what happens after two NPCs talk to each other and what are the conditions for their communication to end.

**Model components**

This model can be decomposed into multiple parts, each of them contributing to the entire process of communication between NPCs.

1. **Entity**: When an NPC does a certain action in the world he sets in motion an ‘event’. We consider an event as any action caused by the player (or the world itself) that can be interpreted by an Entity, allowing that entity to create a message and spread it to other NPCs. Entities are all objects in the game that are capable of reacting to events and because NPCs can be capable of reacting to events they can also be considered entities.

2. **Messages**: Messages are packets of information detailing an event that happened in the world. Messages are the way by which NPCs can communicate with each other. They are the base component of information and give a way for NPCs to interpret events and share them between themselves. The structure of a message can be seen on the Table 1.
(a) **ID:** Every event is associated with an unique ID. A message always knows from what event it was created.

(b) **Transmission Time:** Every time two NPCs share a message, it takes time to transmit that message. Messages take a variable amount of time to be transmitted between NPCs. It is the designer choice when creating elements how much time would it take to transmit certain message if an entity was to release a message to all NPCs close enough to receive it.

(c) **Decay:** Messages lose value over time. Old messages are less interesting to NPCs that new messages, and if two messages have the same content, NPCs will choose the one with less decay.

(d) **TAGS:** TAGs are the result of converting a message description of an event into variables used when two NPCs are deciding what message they are going to talk about.

(e) **Description:** This is the actual text describing the event that happened. When two NPCs talk, the designer can choose to show to the player what are NPCs saying to each other.

3. **NPCs:** NPCs are all game objects capable of receiving and sharing messages between themselves. The can also be considered entities if they are able to create messages as a result of some action being done to them by the player or the world. NPCs have different properties worth mentioning:

(a) **Memory:** NPCs have a finite amount of memory. This means that NPCs can only store a limited number of messages they receive. When a new and more interesting messages is received, it is the responsibility of that NPC to choose the less interesting message to discard from memory in order for the new message to be saved in memory (see Algorithm 3).

(b) **Assertiveness and Cooperativeness:** The desire an NPC has to send or receive a message is measured through this two variables. Assertiveness measures how much an NPC wants to share a message he possesses with another NPC. Cooperativeness measures how much an NPC is willing to receive a message another NPC has to give him. If an NPC has an assertiveness high enough to send a message and another NPC has a cooperativeness high enough to receive it, the next step is to decide if there is something to talk about. The NPC that is sending a message is losing assertiveness over time and the NPC who is receiving that message is loosing cooperativeness. Communication can happen in both directions, since an NPC being assertive has no influence in his cooperativeness value. This way, after an NPC has been assertive, he can switch roles and be cooperative to a new message.

(c) **Interests:** Just as it happens with people, NPCs have different interests from one another. This interests are going to determine how willing they are to receive a new message. If, for example, an NPC has a strong interest in WOOD, he will be more willing to receive a message from another NPC that contains the TAG WOOD. However, if he is not interested in the TAGS of a certain message there is a strong possibility that he will prioritize an old message from memory instead of the new message.

### Final Prototype Concept

When deciding how the final map would be, there were multiple things to consider:

1. We needed a map big enough for the player to understand over time the changes on the map. A big map would also allow for the player to explore the surroundings and increase the change of that player to see interesting and emergent behaviors happening in the world.

2. We needed to give the player an objective. Whatever that objective might have been, it needed to take enough time for the player to understand how the world worked but not long enough that he would take a lot of time to complete it and eventually give up. A longer objective to finish would imply that player testing would take a longer time. The average player testing should take an average of 15-20 minutes.

3. Give the player a sense of progression, so he saw the world change as he was playing the game. Making the game more difficult over time was a strategy used to make the game more challenging and keeping the player more focused.

4. Make the world diverse and with enough terrain obstacles to overcome. Here we wanted the player to move through certain parts of the map where it was harder to reach the objective.

5. The player should also be free to explore the map and interact with the environment.
6. Playing the prototype a second time would end up in players making better decisions and the results would be very different that those with players who never tried the prototype before. So we wanted to make sure players got to explore the map

7. Make the player return to places he has been before. If some important event had already happened there, the player should be able to see the changes in its environment. The player should also be able to notice changes caused by his actions in the behavior of NPCs.

**Implementation**

When implementing a game prototype with player testing in mind, a small story was created, and a map was developed to accommodate that story. The prototype also included a small tutorial explaining to the player what type of actions could he do, and the objective he was supposed to accomplish.

The objective is a simple “Gathering Quest”. The player is asked by “The Prophet” to grab 12 different objects (called ‘Relics’) and to bring them to a certain location. Special NPCs called ‘Guardians’ protect those objects and will chase the player if he grabs them.

The player is only able to carry 2 relics at a time before he is forced to go back to the center of the map (in a place called “The Temple”) and drop them there. The choice of limiting the player carrying capacity to 2 objects forces the player to return to certain locations where there is more than 2 objects to collect, giving the player a chance to observe the changes in that location as a consequence of his actions.

To further illustrate how the gameplay works we created a short gameplay video of the player interaction with our prototype. 4

**Unity Game Engine**

The development of this work was carried out with the Unity™ game engine 5 (version 2017.3.1f1). The flexibility of this engine allowed us to develop all the tools required for the various iterations of the model.

**Tool Development**

Some of these tools were used early on and some tools were used on later iterations. All of them were necessary at one point or another and the final prototype still supports the use of most of them.

**Map Editor**

This tools was the first tool developed and was used throughout the entire development of the model (see Figure 5). This tools allowed us to draw a map in real time for usage in specific scenarios. It supported the use of as many textures as the designer wanted and was possible to draw two different layers of terrain (the foreground layer when holding the left mouse button and the background layer when holding the right mouse button).

**Message Injection**

This tool was capable of giving specific messages to random NPCs at specific times in the game. Every scenario was created with a message sequence and it was possible to repeat a single scenario as many times as needed and try to see patterns emerging. It was possible to shuffle message sequences, determine what types of messages were being procedurally generated and the interval between each message being released on the world.

**Event Spawner**

Message injection was not the only way to test messages in the different simulations. Besides the actions done by the player, we could also want to make an action happen when an NPC passed through a certain part of the map. To archive this a new type of game entity was created called the Event Spawner.

The Event Spawner worked like a trap. When an NPC passed on top of the spawner, he would give the message he was holding to that specific NPC. This was an important tool in initializing certain NPCs with certain messages, as well as testing how the model prioritizes old/new messages.

**Map Loading and Scenario Tests**

The terrain maker was the most important tool to save time making different scenarios, but the most important tool that allowed us to test the model in different situations was the ability to save entire scenarios and load them afterwards. There were multiple components needed to save a Scenario and when the game was loaded it was important that all this components were present. That way there was no time lost in “setting up the experiment” when testing the effect of a variable on an outcome.

As an example of how the map loader works, I want the reader to take a look at Figure 6. Here we can see the multiple parts that compose our model being created and saved in real time. Different entities, terrain layers, NPCs and players are able to be saved and loaded for later use.

---

4. [https://www.youtube.com/watch?v=2jzMMSUyryxU](https://www.youtube.com/watch?v=2jzMMSUyryxU)
5. [https://unity3d.com/](https://unity3d.com/)
Simulation Outputs

The first data outputs registered on the simulations was done through text. We logged not only what messages were being talked and at what time, but also the ratio of repeated/new messages and how many messages were able to reach different NPCs (as we can see on Figure 7).

This type of data was very useful in determining not only if the different parameters in our model were working correctly but also to see what happens in extreme cases, where we can have situations where only X number of NPCs can talk and Y number of NPCs can listen.

This type of data was also very important in determining how the movement and specific location of NPCs around the map can play an impact on the capacity of a message to remain alive in the system (also known as message survivability).

Figure 7: Here we can see, for each message, if it is present and alive in the memory of NPCs. The X-Axis represents the time in minutes and the Y-Axis represents the number of messages exchanged between NPCs during that minute.

Later on we started using Data Graphs to better analyze how messages were being propagated. By this stage we already had an almost complete version of the final map and because of this we were able to record what was happening on specific parts of the map.

We could also get important data like the graph of messages that were alive ('alive' messages were still not at maximum decay) or how many different messages existed in the world. (see Figure 7)

Special Considerations

There were many things that were considered before implementing the final model. As a result of extensive testing during development, some decisions had to be made and some parts of the model had to be adapted. The special considerations described below should be very helpful to any person that tries to implement this model.

Predictability of Encounters

Since the designer doesn’t know where NPCs are going to be at an exact time (because they could, for example, have stopped to talk to other NPCs) he can’t know how a specific message he wants to test is going to spread across the network.

Even with the unpredictable nature of the NPC’s state in the world (where are they going to be at a certain time), it was possible to determine that the rate at which messages were being talked about remained very similar in multiple tests using the same message sequences.

Expected vs Unexpected Behaviors

Even when running the same scenarios over and over again, different results can sometimes be obtained. This happens because our prototype calculates the distance between NPCs every frame and if 3 NPCs are at a distance between themselves that allows for communication to happen, there is no way to know who talks to who. Is a certain situation NPC_A could be talking first to NPC_B and in another one he could be talking first to NPC_C.

Even with the unpredictable nature of the NPCs in the world (where they are going to be at a certain time), it was possible to arrive at some common conclusions about the behavior of NPCs in that specific scenario when repeating the simulation multiple times.

NPC Interaction and Proximity of Events

The distance required for two NPCs to start talking can change a lot and will end up being a design decision that has a big impact on the game when applying our model. The further away the designer allows two NPCs to communicate with one another, the higher is becomes the chance of a message being communicated between them, since they can talk over long distances.

If the distance at which NPCs can see events happening in the game world is too small, there will be a lot less messages being transmitted between NPCs. If the distance is too big, a lot of NPCs will see that event and receive a message. If all NPCs in the map can see the same event, they will have a lot less interest in talking about that even.

So, "The higher the number of NPCs that know of an event is, the lower is the chance of that event being talked about", since NPCs will always prioritize a message one of them doesn’t know instead of a message both of them know.

Conversation Duration

The duration of a conversation not only affects how much time it takes for an NPC to transmit a message to another NPC, but also how the values of Assertiveness and Cooperativeness increase and decrease over time.

This is one of those variables that only through testing and manipulation a designer make sure that the rate of transmission of messages is happening the way he wants.

Since NPCs cant receive messages from other NPCs not involved on the conversation, if messages take too long to transmit there will be less messages circulating on the network and as a result the player might have to wait for a certain message to reach the NPC he wants.
If conversations are too fast, the rate at which MessageA reaches LocationA will depend primarily on the movement speed of the NPC who has that message and to what destination is he moving to.

If messages take no time to transmit, NPCs will not have time to decrease their values of Assertiveness and Cooperativeness. Because of this, they will keep talking forever, never moving from their position or changing their behavior. One way to fix this is to lower the values of Assertiveness and Cooperativeness instantly after each conversation, instead of decreasing this values over time.

Evaluation

To test our model, two different versions of the prototype were elaborated:

1. **Version A**: This version did not make use of our NPC Communication Model for sharing messages. Instead it uses an alternate version of our model where messages are transmitted to all NPCs in the map instantly. This behavior was used in order to simulate the behavior of games that typically use a centralized system for how information is gathered and shared among NPCs.

2. **Version B**: We used our model to shape the behavior of NPCs. This time, instead of the player’s actions being sent as messages instantly to all NPCs, those messages propagate over time in a network of NPCs, each with its own interests.

The first version did not use our model and as such, every time the player grabbed an object to get closer to the final quest goal, the NPCs involved in chasing the player would know what the player’s actions were. This was the main problem our model was trying to solve.

On the second version of the prototype we used our model to shape the behavior of NPCs. This time, instead of the player’s actions being sent as messages instantly to all NPCs, those messages were spread over time.

Data collection methods

There were two different data collection methods used to reach a conclusion regarding how viable a model like this one would be useful in video games.

The first method was through the use of a simple questionnaire with the feedback from the player regarding the play through of each version of the prototype. Using a questionnaire, we wanted to observe if our model increased the gaming experience of the player and if the player was able to understand how NPCs were communicating between themselves.

The second type of data collected was in the form of a database, where we were recording what the player was doing in different parts of the game. This type of data was important in having a deeper look at how the player played the game in each version. We will describe in more detail each data value captured in the section below.

Questionnaire Data

For this evaluation, we asked participants to play both versions of the prototype, with players randomly choosing what version they wanted to play first. At the end of playing each version each player was asked two questions:

1. Rate his overall gaming experience in a scale of 1 to 7.
2. How credible it was the communication between NPCs, meaning how easy it was for the player to understand what type of things NPCs were talking about. This was also a rating on a scale of 1 to 7.

If the players gave different ratings on these 2 questions in each version we would then ask why they rated differently. This feedback was important in knowing if our model worked correctly and if not what could be done to further make it better in the future.

Game Data Collection

Each time a player tried one of the versions, a few values were being sent to a database. This values were:

1. **Relics Grabbed**: We not only knew what relic was grabbed by the player but at what time was that relic grabbed. This was useful in seeing if the player was getting better at the game and, with increased difficulty, if the player took more time in grabbing the next relic.
2. **Grabbed by Guardians**: We also wanted to see how many times the player has been grabbed by guardians and how he adapted to the game, managing to escape being chased by NPCs. If the player was dying a lot in later stages it could mean that the game was being too difficult and if he died a lot more at the start it could mean that the player was having difficulties with the way he controlled the player.
3. **Tutorial Time**: We wanted also to see how much time each player took to finish the tutorial. This was a good indication of how at ease the player was with the controls and also show how experienced the player was at playing this type of games.
4. **Game Time**: With this data we could see how much time the player took to finish each version of the game and see if the model made it easier for the player to find all game objects or if there was no substantial difference in the time it took to finish the game.

Player Testing

We hypothesize that the overall game experience of the player will be significantly better on the version with the model as a result of the communication happening between NPCs. The question relating to the credibility of the communication happening in the world should also be significantly better on the game version with our model present. The ability of the player to complete the objective should not be different on both versions.

The version of the game with our model should also lead to a decrease in objective completion time since we assume that with our model the communication happening in the world would be more believable and play an essential role in the conclusion of the objective.

To compare the different versions of our prototype we used a Wilcoxon signed-rank Test to test game experience and NPC communication with our participants. Both variables tested were ordinal variables on a 7-point Likert scale, where we tested both pairs of questions with both prototypes.

Sample

This evaluation was done with 15 people, with ages ranging from 9 to 32 years old (M = 22.33, SD = 6.332). On the pie charts of Figure 8 we can see the different types of users that tested our prototypes.

Procedure

Some of the tests were done face-to-face with our participants and other ones were done remotely. All participants were asked to perform the following tasks:
First each player was asked to respond to 4 demographic questions: age, sex, gameplay frequency and familiarity with the game genre. After responding to this demographic questions, each participant was asked to play Version_A or Version_B at random. At the end of playing each version, the participant would rate his overall gaming experience and how well did he understand the communication between different NPCs. They were allowed after playing both version to change the ratings they gave to both questions. They were also asked the reasons for the difference in the given ratings.

**Results**

![Figure 8](image)

*Figure 8: In this figure we can observe that 86.7% of participants had experience in playing video games and that 60% of them already played at least 1 game similar to the one on the prototype.*

As we can see in Figure 9, the mean value of the gameplay experience of the players using both versions was Version_A = 4.13 and Version_B = 5.2. A Wilcoxon signed-rank test showed a statistical significant difference regarding the overall gaming experience when using both models \( Z = -2.115, p < 0.05 \). This leads us to conclude that a version using our model resulted in an increase in overall gameplay experience of the player, as we expected to happen.

When we talk about the ability of the player to understand how NPCs communicate in the world, there was a significant difference in perception from the players since we have mean values of Version_A = 3.2 and Version_B = 6.33. A Wilcoxon signed-rank test showed that the use of our NPC communication Model had a statistically significant impact on the way players understood the communication happening between NPCs \( Z = -3.453, p < 0.01 \). This suggests that the communication between NPCs using our model is a much more desirable feature than a version without it.

We can observe here that the average play time using our model was 10 minutes and 9 seconds in comparison to the 14 minutes and 21 seconds taken to complete the version without our model. This supports the hypothesis that our model helped the player complete his objective faster without compromising his gameplay experience.

![Figure 10](image)

*Figure 10: Here we summarize the test results, and observe a positive impact of our model on our game.*

Another possible reason of a significant difference in gameplay time could also be an increase in difficulty on the version without our model, since all guardians are after the player the moment the player grabs the first relic. Even with an increase in difficulty, the Wilcoxon test showed no statistical significant difference in the number of times players were able to complete the objective \( p = 0.18 \).

The Wilcoxon signed-rank test (see Figure 10) also shows us values of \( p < 0.05 \) for the time it took the player to complete the objective, so there is a statistical significant impact of our model on the prototype in this regard. The player could have taken less time in completing the objective using our model because not all NPCs knew the player’s actions and as a result only some of them chased the player. This behavior could have helped the player have a better reading of what was happening in the game and made him more efficient in the conclusion of the objective.

**Player Feedback**

We also were able to gather of player feedback from our questionnaire. Many players said they were able to see their actions reflected on the NPCs behavior. Some players also noticed that when they gathered a relic, one of the NPCs that was close by moved towards the guardian to report that player’s actions. Since that point they avoided grabbing relics when there were NPCs close by that could catch them in the act. Some players also reported that guardians came after them after they had gathered a relic. Those players clearly understood that they should not grab relics when they were in perception distance of guardians.

One of the participants commented: On the first version (without the model) I understood what they (the guardians) were talking about but I did not understand the purpose of their conversation. On the second version (with our model) I noticed that when I broke a relic one of the NPCs was going to tell a guardian that I had done and after this he started following me. One of the times, when I was coming back for the second time to the top part of the map, the guardians had changed and were now looking for me.

Another participant told us: On version B (with our model) the ghosts (guardians) seemed more capable of communicating with other NPCs that on version A (without our model). On this version (Version_B) they also seemed to understand what the player was catching and not just that the player wanted to catch

This kind of feedback was extremely important for us to understand our participants reason for giving different ratings for both versions and understand better the impact our model had on their gameplay experience.

**Conclusion**

With this work we presented a model for how NPCs in a simulated world environment could communicate between themselves in a more believable way. This type of communication would be not
only important for a possible change in behavior from the NPCs but for the player to see that the world he was experiencing was more complex and believable. Our model was able to describe the world by different types of parts and explain how those parts fit together to make a different and more believable type of communication happen between NPCs.

We started by researching how current games try to change the behavior of NPCs to make the player more engaged and interested in the world and the story. Like the games described in the Related Work section, our model offers one more tool for increasing the complexity of the game world and the NPCs who inhabit that world, giving the player an overall better gaming experience.

It is also quite important to highlight the incremental development of different types of tools, all of them created at a certain point to test our model and how did he behave under stress. This tools also allowed us to play around with different model parameters and observe how the world reacted to them. Our model was tested over and over again, perfected until we reached a working version described in this document.

Our approach was centered around the player, with the focus being on the gameplay experience a player would have if he was to play a system with our model implemented. Each incremental step we asked ourselves: "How does this change make the game better?" and "What different possibilities can emerge if we make this changes?"

We then started creating simple gameplay scenarios and as the different maps became more complex, so did the behavior of our NPCs. From just 20 NPCs to start with and expanded the map until we had a finished prototype with over 100 NPCs, all moving around the map sharing sharing messages between themselves. Statistical data was collected during every stage of development and this data was worked upon until the values we were getting were in line with the ones we wanted.

The elaboration of tests with human players, where they were asked to play two different versions of the game, one using our model and the other using a standard approach to how information is shared in current video games was the final way by which we determined if our model was functional.

We had very positive results from our participants in the final testing phase. Participants were not only able to understand what the NPCs were talking about, but also why they were talking about it. They were able to adapt their gameplay to the communication happening between NPCs and as a result changed their behavior. That way, NPCs that lived very far away could end up living close to one another after some time. Participants and the other using a standard approach to how information is shared in current video games was the final way by which we determined if our model was functional.

We believe that our model is a step forward in making more realistic worlds for the player to explore, increasing the replayability of the game and possibly enabling emergent gameplay situations to occur.

**Future Work**

There are several ways of continuing the work executed throughout this project, but we are just going to point out the most important ones. Some components that were excluded during the development of the final prototype (for the sake of simplicity and saving time) but could be implemented in further developments of this model.

They are: Friendship Level, Goal Oriented Movement and Level of Detail

This components can not only make the communication between NPCs seem more believable, how they behave in the presence of each other and also how they move and make decisions in the world.

**Friendship Level**

Although "friendships" are not present in the final model, the decision to not include this variable was in order to keep the model simple and without too many variables that could break the behavior of NPCs or make the model more difficult to evaluate.

Friends have a stronger chance of talking to each other. When determining a value of each message two NPCs are going to talk about, you have to take into consideration that if the value of all messages is too low and does not pass a minimum threshold, the game designer could choose to prevent NPCs from talking completely. Even if there is a message they could talk about, that message is not important enough for those NPCs to stop what they were doing before.

In previous versions of the model there was the possibility of mapping the friendship level between two NPCs and when considering the most attractive message to tell another NPC, the total value of a message was multiplied by the friendship level of the relation between both NPCs.

This is not an easy parameter to balance and can possible break the gameplay experience of the player. Further solutions to this component could be explored as the model is currently able to support the usage of different friendship levels between NPCs.

**Goal Oriented Movement**

As described above when talking about "Smart Terrain", it would be very interesting to know how information would be shared if NPCs moved based on their personal goals instead of a fixed pre-established movement. Some locations would naturally become more populated, and some locations could end up without NPCs.

The communication happening in the world has an effect on NPCs and their goals changed over time, so would their movement change. That way, NPCs that lived very far away could end up living close to one another after some time.

**Level Of Detail**

A possible future work here would be to make different levels of detail for different types of NPCs. There are many ways of accomplishing this:

1. Disable NPCs that are further away from the player and give more power to NPCs that are closer to the player and on vision of the camera.
2. Give NPCs near the player the power to use path finding techniques and obstacle avoidance and remove power from NPCs that are further away, giving them a simple patrol type of movement.
3. Allow for NPCs that are near the camera to work properly with path finding. NPCs that are further away would only be active based on the importance of their messages.

Option number 1 and option number 2 are very straightforward to understand but option number 3 is not so easy to understand.

As an example for option 3, lets say that \( N_{PC_A} \) is very far away from the player. Calculating the distance between the player \( N_{PC_A} \) the game reaches the conclusion that only NPCs that have messages with \( LevelOfImportance > 4 \) would be enabled. Since \( N_{PC_A} \) has a message with \( LevelOfImportance = 6 \), he is able to share that message with disabled NPCs around him. Notice here that even if an NPC is disabled, he must still be able to receive messages from enabled NPCs around him. There is no point in having an enabled NPC far away from the player if all the rest of the NPCs cannot receive that important message.

**References**

