A Merchant Model

Improving player experience with NPC vendors in Role Playing Games

Miguel Oliveira Ferreira

Thesis to obtain the Master of Science Degree in

Computer Science Engineering

Supervisors: Prof. Pedro Alexandre Simões dos Santos

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Pedro Alexandre Simões dos Santos
Members of the Committee: Prof. Rui Filipe Fernandes Prada

October 2017
Acknowledgments

I want to thank my advisor for given me the support and monitoring needed to finish a work of this magnitude.

To my family: my mother, my brother and my sister, I want to thank as well for all the support, comprehension and for always being there, more than happy to help in any way possible.

Another person that was always there was Rita, I couldn’t finish this without her, probably couldn’t start either. By supporting one another throughout all this time, we managed to finish our objectives. Above all I knew i could count on her without any doubt, again, thank you for everything.

Next I want to thank my friends, Diogo, Manuel and João for a whole year of constant help. Hopefully now, Mara Studios will be stronger than ever.

I want to mention Antonio as well, for passing for all the trials and tribulations with me in this work, we did it. Finally, I couldn’t do this without the Institution itself and all the resources provided to me.

To each and every one of you – Thank you.
Abstract

In modern video-games, a lot of work and money is invested in increasing the immersion of the player in the game. Areas like graphics, physics and story telling are improving each year to become more lifelike. Unfortunately, the same effort has not been so prominent in the improvement of believability on relationships between characters. Our work is focused on improving the interactions between the player and merchants in games. Currently, in most video-games, your actions, as the player, don’t affect merchants behavior or inventory in any way. We studied and used principles of existing social architectures and knowledge gathered of merchant behavior in order to create a generic model that can be applied to video-games to enhance the player’s experience in terms of immersion when interacting with said merchants, always minimizing the authoring effort.

An abbreviated version of the model designed was implemented and tested in the game Conan Exiles using his modding tool.

The results of these tests were very positive, our model proved to be significantly more enjoyable, believable when compared with a merchant without our model and not compromising the flow of the interaction between the player and the merchant. Also, there was an almost unanimous preference for the merchants that had our model in relation to the one without it.

Keywords

merchant; social; video-games; Conan Exiles
Resumo

Nos videojogos actuais, muito trabalho e dinheiro é investido no aumento da imersão do jogador no jogo. Em departamentos como os gráficos, física e narrativa melhoram de ano para ano na tentativa de se tornarem o mais reais possível. Infelizmente, o mesmo esforço não se verifica na credibilidade das relações entre personagens. O nosso trabalho foca-se na melhoria das interacções entre o jogador e vendedores em jogos. Actualmente, na maioria dos jogos, as acções de um jogador, em nada afectam o comportamento dos vendedores ou o inventário que vendem.

Nós estudamos e usamos princípios de arquitecturas sociais existentes e comportamento de vendedores de forma a criar um modelo genérico que pode ser aplicado a videojogos para melhorar a experiência do jogador em termos de imersão quando interagindo com esses vendedores, tendo sempre em conta a quantidade de autoria necessária para a criação deles.

Uma versão abreviada do modelo criado foi implementada e testada no jogo Conan Exiles usando a ferramenta de modding deste.

Os resultados destes testes foram bastante positivos, vendedores com o nosso modelo provaram ser significativamente mais agradáveis e credíveis em relação ao vendedor sem o modelo, e também se verificou que não existiu compromisso de “flow” nas interacções. Também se verificou uma quase unanimidade na preferência dos vendedores com o modelo implementado em relação ao vendedor sem ele.

Palavras Chave

vendedor; social; videojogos; Conan Exiles
## Contents

1 Introduction .................................................... 1  
   1.1 Problem .................................................. 2  
   1.2 Objectives ................................................ 3  
   1.3 Contributions ............................................ 3  
   1.4 Thesis Outline .......................................... 4  

2 Related Work .................................................. 5  
   2.1 Trading techniques ....................................... 6  
      2.1.1 Door-In-The-Face: .................................. 6  
      2.1.2 Low-Ball: .......................................... 7  
      2.1.3 That's-Not-All: .................................... 7  
   2.2 Partner Selection .......................................... 8  
   2.3 Trust influence on sales .................................. 9  
   2.4 Psychological models and definitions .................... 9  
      2.4.1 Affective and cognitive empathy: .................. 10  
      2.4.2 Friendship Model: .................................. 10  
   2.5 State Of The Art: ......................................... 11  
      2.5.1 FAtiMA .............................................. 12  
      2.5.2 Comme il Faut (CiF) ................................ 13  
         2.5.2.A Prom Week ...................................... 14  
         2.5.2.B Comme il Faut - Creation Kit (CiF-CK) .... 16  
      2.5.3 PsychSim ............................................ 17  
         2.5.3.A SocRATES ....................................... 19  
         2.5.3.B Thespian: ....................................... 20  
         2.5.3.C BiLAT: .......................................... 20  
      2.5.4 The Friendly Blacksmith ............................. 20  
      2.5.5 Misrepresentation Game .............................. 22  
   2.6 Games ..................................................... 23
List of Figures

2.1 Classifying Potential Partners [1] .................................................. 8
2.2 Friendship Model [2] ................................................................. 11
2.3 FATIMA Core [3] ................................................................. 12
2.4 Prom Week [4] ................................................................. 15
2.5 PsychSim Interface [5] ....................................................... 18
2.7 Bilat game screen [7] ................................................................. 21
2.8 Fixed-pie lie example [8] .................................................. 22
2.9 Normal merchant interaction in Witcher 3 ....................................... 23
2.10 Animal Crossing Shop ................................................................. 25
2.11 This Merchant Life ................................................................. 25
2.12 Legend of Zelda ................................................................. 26
2.13 Mammoth chickens mod: Adds giant chickens to the video-game Skyrim [9] .................................................. 27
2.14 Conan Exiles DevKit launcher ........................................... 28

3.1 Proposed merchant model ................................................................. 33
3.2 Merchant Model Example ................................................................. 37
3.3 State machine of the dialogue tree system, where the criteria of the tree selection is the relationship value being above or below 5 (this value is customizable) .................................................. 38
3.4 Dialogue tree used on a merchant .................................................. 39
3.5 Full set of dialogue trees existent in a merchant .................................................. 40

4.1 Implemented model ................................................................. 44
4.2 Normal state of dialogue ................................................................. 47
4.3 Merchant expecting an offer ................................................................. 47
4.4 UI of the shop with instructions for the testers .................................................. 48
4.5 Relationship Structure ................................................................. 49
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPC</td>
<td>non-playable character</td>
</tr>
<tr>
<td>RPG</td>
<td>role-playing game</td>
</tr>
<tr>
<td>CIF</td>
<td>Comme il Faut</td>
</tr>
<tr>
<td>CIF-CK</td>
<td>Comme il Faut - Creation Kit</td>
</tr>
<tr>
<td>UI</td>
<td>user interface</td>
</tr>
<tr>
<td>FAtiMA</td>
<td>Fearnot AffecTlve Mind Architecture</td>
</tr>
</tbody>
</table>
# Introduction

This chapter introduces the topic of the thesis, explaining the problem, objectives, and contributions. It also outlines the structure of the thesis.

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Problem</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Thesis Outline</td>
<td>4</td>
</tr>
</tbody>
</table>
Monetary retail transactions exist since ancient times, and as such, they evolved through time until reaching the state we now know, with online shops and huge shopping malls being the go-to place when someone wants to buy something. A product of the evolution of transactions was the implementation of special deals and promotions from time to time, depending on several factors (time of the year, stock etc.). It is evident that transactions are much more than a simple trade of ‘x’ currency for ‘y’ product, there is a lot more to take into account. Every time we physically buy something from someone, we have a wide range of possible interactions, from the simple greetings and transaction, to the bargain of the price or the evaluation of current promotions.

The same transactions can be seen, not only in the real world, but also in video-games, mainly due to the fact that these want to simulate certain aspects of the real world. This can be seen more frequently in some genres of games like the role-playing games (RPGs). In these games the act of selling and buying products is very common, but, despite this, there is a clear lack of depth in this process, feeling, more often than not, like a mechanized process.

Current video-games can have an enormous amount of money invested in them. For example, the game “Witcher 3, Wild Hunt”, reported a development cost of 46 million dollars, while “Grand Theft Auto V” reported a development cost of 137 million dollar. Despite these investments, the industry prioritizes the allocation of that money into the engine, world, graphics and authoring, leaving the artificial intelligence and social interactions components behind.

As a result of this, for example, the interactions with merchants in this kind of games can feel unrealistic and, most of them, feel like a simple means to do transactions instead of a simulation of a real vendor who is trying to make a living and may empathize or have a bad impression of the player.

The degree to which the player feels integrated with the game space is a measure of her or his sense of “immersion” [10], and there are a lot of recent games who accomplish this in several components, but most of the immersion generated by the beautiful worlds and quests may fall flat when the player interacts with a non-playable character (NPC), for example, with merchants.

From here, we will refer every seller, salesman and vendor as merchant, to keep consistency throughout the paper and because it is the most used name in video-games to represent an NPC that sells or buys goods.

1.1 Problem

Most recent RPGs have some kind of merchant or trading system implemented, it can be a character, a machine or a simple user interface (UI) element that presents the player with the option to buy items, upgrades or basically anything that the developers of the game want.

The problem is that, one of the main goals of any RPG is to immerse the player in his world, and to
do so, every component of the world itself has to be believable enough to make the player feel that he
belongs there. Most part of those components have evolved throughout recent years, but the interactions
with NPCs, more specific with merchants, is something that, for the major part, is still limited to a simple
exchange of currency for product.

This makes it so the experience of a player in one of these games, can be negatively affected by the
lack of believability in the merchants that populate said games. Because the immersion of the player is
always as high as the least immersive component he interacts with, the interactions with the merchants
without a social component can produce a bottleneck effect in the overall experience.

1.2 Objectives

The motivation behind this work is to improve the experience of the player within an RPG by creating
a model that, when applied to merchants in that game, increases the range of interactions possible,
honing the believability of said merchants and balances the authoring effort of the developers.

The model created allows the player to have the possibility of having not only more interactions with
the merchant, but making those interactions as realistic as possible but always balancing with the game
experience itself.

With is, we recognize that, above all, the merchants that currently exist in video-games are there for
convenient reasons, that is, to make it easy for the player to buy or sell items.

We value this practicality and it’s part of our goals to keep the flow of the interactions as simple
and straight forward as possible because, if we only had the realism and believability in mind without
taking into account elements of flow and convenience, we could argue that it was a good idea to make
the player wait for hours before getting to talk to the merchant because the shop was full of people in line.

On top of this, it is important to minimize the amount of authoring that the developers have to do
when implementing our model. The development of RPGs already brings with it an enormous amount
of authoring (with the ever increasing dialogue trees or quest lines present in these games), taking this
into account, it is relevant for this work to not increase this authoring, and, if possible, prevent part of it.

1.3 Contributions

The work presented in this paper brings out really interesting conclusions, as we will see in the results
chapter, but, despite that, we can say that every step leading to that conclusion has something that is
worth retaining.

This thesis begins with a study and presentation of the several areas regarding transactions and
trading, psychological and social models and the state of the art of social architectures and other works relevant to the project.

After gathering the necessary information, a theoretical model was designed with what was learned from the areas studied. The goal of this model was to meet everything stated in the previous section but keeping it generic enough to be possible to be integrated in any project or video-game.

An abbreviated version of the model designed was implemented in a current commercial video-game, this version of the model, despite being an abbreviation, has the required components to produce the same results as the one proposed in Chapter 2.

We also did a validation of the model created, with positive results, as can be seen in chapter 5. The results show us that players tend to have a better experience when interacting with merchants that contain our model. And the authoring done was less than necessary if we would want to replicate the same interactions without the model.

These were the main contributions done by the core of our work, but we also studied and implemented a working dialogue system that was lacking in the game where the model was applied, this was a product of necessity but the end result was positive and functional.

With all of this, we hope the work done can represent a contribution to the areas of development, immersion and sociability in video-games. Helping to improve the experience of the players while playing the game and of the creators while making said game.

1.4 Thesis Outline

This document is organized as follows. The second chapter details the areas studied in order to design our model, it includes strategies of trading, social and psychology models, recent video-games relevant for the inclusion of merchants that are lacking in areas we want to address, and some social architectures researchers by academic groups. With this chapter we want to conclude which components our model should have and in which game it can be implemented to validate said model.

Chapter 3 talks about the model implemented, the result of all that was studied in chapter 2. In there we detail the model theoretically and address other necessary tools created to support our model.

In Chapter 4 we go in specifications about the implementation of the model designed in the previous chapter, justifying every relevant step of the process and explaining it with the help of code snippets.

As for Chapter 5 we validate our model and to show and justify the results of the questionnaires done to the testers. Finally, in Chapter 6, conclusions are drawn and we talk about the future work that can improve this thesis.
Related Work

Contents

2.1 Trading techniques ................................................. 6
2.2 Partner Selection ................................................. 8
2.3 Trust influence on sales ......................................... 9
2.4 Psychological models and definitions ......................... 9
2.5 State Of The Art: .................................................. 11
2.6 Games ............................................................. 23
2.7 Dialogue Systems in Games ..................................... 26
2.8 Modding ............................................................ 27
2.9 Discussion ......................................................... 30
In this Chapter we will emphasize the fact that this work embraces several distinct fields of expertise and that we have the intent of merging knowledge of those areas in order to create the proposed solution.

We will focus on the work that was studied unrelated to computer science. In here, techniques and models are going to be referenced as a base and/or inspiration to the creation of a unique model that can be applied to merchants in video games to make them more believable. The areas studied will be related to marketing, commerce, sales and also with psychology and sociology. After that, we will look into the state-of-the-art in terms of social architectures and other works that we perceive as relevant to our project.

2.1 Trading techniques

Trades can be traced back to prehistoric times, they had been practiced in late paleolithic an early neolithic times [11]. These trades suffered changes along the time. Babylon, Assyria and other ancient civilizations adopted the open air, public markets. These where inhabited by skilled artisans that were selling their products. Eventually, negotiation, marketing and trading techniques were developed by the merchants to make their products and deals more appealing. Those strategies range from simple advertising methods to complex deals backed up by a lot of socioeconomic studies.

A lot of times, deals are designed to give the impression that the clients have an opportunity they won’t get anywhere else, this makes it so that the business that better executes this kind of trading strategies gains an advantage against other business in the same position. As such, these techniques were heavily studied and are really important in order to make successful deals and sales.

There are several techniques that real vendors tend to use in order to promote or facilitate the selling of their goods. Some of these strategies can be applied to NPCs in order to try and make the interactions with them more immersive. We are now going to give three examples of these strategies:

2.1.1 Door-In-The-Face:

This technique works by getting a no, to increase the change of getting a yes after, due to a high contrast perceived by the person who is the target of this technique.

Proposing an extreme request which is rejected and then moving to a smaller request increases compliance with the smaller request. If only asking for the smaller request, the merchant faces less compliance than if opting with the first technique.

According to MLA Cialdini, Robert B., et al. [12] where several experiments were done to understand this technique, it was possible to conclude that by rejecting an initial extreme request, the target “puts himself in a position from which virtually his only possible retreat is accession to the smaller request”. Hence, making the first extreme request is very important for the effectiveness of this technique. If this
does not happen, compliance with a smaller request will not be significantly enhanced. However, this will only work if the second request is considered a concession on the part of the merchant.

This method can be applied by simply proposing to the player an expensive item, and upon the rejection on that transaction, showing a more cheap item.

2.1.2 Low-Ball:

The Low-Ball technique takes advantage of the commitment developed by people after they accept some request in order to keep them accepting other, more costly requests [13].

This can be seen, for example, when a car salesman presents a client with the base price of a car, and, after the client comes to terms with that price, the vendor starts adding extras to the car that make it more expensive.

People, when complied with an initial small request were found to be more likely to agree to a similar but larger request [14]. This effect can be replicated in game by having some items unlocked as soon as you buy other relatable items or by having the merchant saying that for an extra small cost, he can improve the weapon bought by the player.

2.1.3 That’s-Not-All:

That’s-not-all involves the incremental appeal of a deal without giving the client a chance to think or reply to the constant changes. This makes the client see the contrast between the first proposal and the current one and feels an increasing obligation to purchase the product to compensate for the merchant’s concessions [15].

Offering a product at a high price, not allowing the customer to respond for a few seconds, then offering a better deal by either adding another product or lowering the price [15]. A version of this method that is based on the same principle can be applied by having special prices disappear when the player closes the inventory, forcing the player to choose under pressure.

This technique effectiveness can be explained by two theories. The first one because the merchant negotiates on the purchase price, making the costumer feel obligated to respond to the act by agreeing to the better price. The second, since the original price alters the anchor point against which the purchase decision is made. By lowering the original price or including another product, the costumer feels like he/she may be getting a better deal and also increases the chance of the second price falling into within the range of acceptance created by the new anchor point.

These are some examples of techniques that could be applied in modern RPGs in order to make the merchants more believable and interactive.
2.2 Partner Selection

The importance of being selective when choosing a partner is proved to be very crucial (when applied to electronic commerce). Firms that are more selective with their partners are more prone to have healthier businesses [16]. A model that helps describe what are the attributes that characterize a potential partner to do business with, was created by D. Wilson in 1995 [1]. This model presents the best partner as the one with the lowest risk to make business with, while adding a great value to the product (as seen in Fig. 2.1). The main goal is to balance risk with value when choosing a business partner.

![Figure 2.1: Classifying Potential Partners](image)

It is not illogical to think that an adaptation of this model could be of high value when applied to RPGs, with a change of variables to adapt the model to the context of the game. This model has the potential to open the door for a whole new range of interactions and possible relationship beginnings that present a new angle of approach between the merchant and the player.

The partner selection model presented here reflects a very pragmatic approach to the topic. If something like this was ever implemented in the model developed, the “human” factor had to be weighted in. This is, if it was to be believable, a component correspondent to a personality match between the client and the merchant should be taken into account on top of what was discussed in this section.

This would, hopefully, make the interaction reflect the merchant has an agent with a human side, and a business side.
2.3 Trust influence on sales

Trust has a very significant weight in society and in the interactions that each person does with another. People who do not trust others avoid interacting with them [17].

The amount of trust one person has in relation to another depends on their personalities, their history together, their relationship among other factors. This is something that our model can do and it will be discussed more in depth in the section 6.2.

From a sales standpoint, an increase of trust from the client side either towards the merchant or the firm where that person works, can have very positive effects towards a lot of factors that ultimately improve sales potential. In this context, trust can come from several direct interactions with the salesperson or the reputation of the same.

If the salesperson invokes an above average trust from the client, the tendency of that client to purchase from the salesperson can be up to 64% higher [18].

With all that, it is evident that it’s of the most importance for a merchant to try to gain the client's trust with the right actions and interactions and it is also obvious that, trust is something that is present in everyone, in different ways, but it is something that, if applied to an agent, can make it more believable and rich for the player to interact with.

2.4 Psychological models and definitions

As mentioned in the section 1.2, one of the main objectives of this work is to make a merchant and the interactions with him more believable. This means that we have to study some psychological models so that the merchants’ behavior can hopefully, mimic as well as possible, the one on a normal person. As such, this section shows some models that can be relevant to be implemented in the model.

Every person is different, as such, everything discussed in this section will be seen as something that can vary from merchant to merchant. Because everyone is different and we want to transpose that to our model.

With that said, in this section, a variety of models describing empathy and friendship will be referenced to serve as basis for some behavior and interactions of the merchant NPC. Ideally, it would help determine how the merchant should behave not only in sporadic situations, but also in a scenario of a relationship.
2.4.1 Affective and cognitive empathy:

Empathy can be defined as an affective response that comes from the comprehension of another's emotional state or condition [19].

In the day-to-day interactions, empathy is what makes a person seem nicer and more comprehensive (or more non-friendly). It is extremely important because, depending on our actions, relationships with people around us are developed in different ways, and since one of the factors that influence our actions is empathy, it is clear the importance of it.

Empathy can affect directly the response to a situation, but, maybe more important than that, it can affect indirectly a relationship as a whole. People tend to want to become closer to people who have higher levels of empathy, as such, along several interactions, different values of empathy can result in different kinds of relationships.

Generally, there are two types of empathy accepted when talking about this subject [20]:

- Affective empathy, related to the ability to experience and share the emotions of others.
- Cognitive empathy, related to the ability to understand the emotions of others.

Studies regarding bullying support that, a low cognitive empathy can be the cause of indirect (not physical) violence, like social exclusion and gossiping while a low affective empathy can be responsible for more direct and violent actions [21]. This can be applied not only to bullying but to other forms of negative relationships.

With this information we should be able to define more easily an empathy factor to the merchant NPC, how can it vary and the consequences of those variations. It is relevant to make the model in a way that allows the creation of merchants with different "personalities" and empathy is a good pointer of that.

2.4.2 Friendship Model:

The study of a friendship model comes from the fact that one of the objective of this work is for the model to allow a relationship between the player and the merchant, that can be developed either positively or negatively and that has some weight in the decision of the outcome of interactions.

As such, a model that sections a friendship development into five stages was studied, in which, each stage represents a level of commitment of said relationship with normal behaviors and mentalities associated to it [2]. Figure 2.2 shows a representation of these stages:

- Strangers: Interactions are very superficial, with little to no personal information revealed.
- Casual Acquaintances: More interactions and conversations. Still no shared personal information but one can talk about his feelings about positive subjects (non-controversial).
• Friends: Friends feel secure enough with each other to be more spontaneous and share some personal stories. More personal and negative subjects come into conversations than in the past stage, but not as much as positive ones.

• Deep Friendships: It takes longer to reach this stage. Deep friends share good and bad experiences.

• Self-intimacy: It's not healthy to share everything that comes to mind. This stage implies that a healthy relationship with others comes from a healthy relationship with yourself first.

This model, may help our work by discriminating which stages of relationship our model has, and what are the characteristics of each one, based on the stages mentioned above. This allows us to implement a threshold system that, depending on a value correspondent to the relationship, lets’ us determine the status of said relationship, and calculate the outputs accordingly.

2.5 State Of The Art:

Here, we present a description of a series of architectures and models, related with computer science, that can be of use to our work. They range from negotiation models to agent architectures and we can take a lot of inspiration and ideas from them. Some will have a direct impact on our work, whereas others can serve has possible future improvements and ways to expand it.
2.5.1 FAtiMA

FAtiMA (Fearnot AffecTIVE Mind Architecture) is an Agent Architecture with planning capabilities designed to use emotions and personality to influence the agent's behavior [3]. This architecture is modular in order to accommodate just the necessary features without compromising simplicity and size. With that said, there’s a possibility of using a lighter version of FAtiMA that uses just enough components to accomplish the task it was given to. This was done to solve the problem that was created when several institutions and scenarios (such as FearNot! [22], ORIENT [23], and a process Model of Empathy [24]) started to implement this architecture, making it scale very fast in terms of compatibility.

The modules themselves are generalized and independent from each other so that they can be applied to a scenario later on and to avoid conflicts between components.

This architecture is composed by a core (FAtiMA Core, Figure 3) that serves as a base for the components to be added later:

![Figure 2.3: FAtiMA Core [3]](image)

When perceptions are received, the agent’s memory is updated and the appraisal process starts. The result of the appraisal process is stored in an affective state and later used to influence the action selected.

The appraisal derivation is responsible for attributing appraisal variables (like desirability) according to the input (perceptions).

The affect derivation takes into account the appraisal variables and an appraisal theory and gener-
ates an affective state (mood). Just the core by itself doesn’t do anything, it needs components for it to work. These components will, one by one, add features and characteristics to the agent (either by adding appraisal variables that translate to emotions, creating a model that has the internal states of other agents, determinate limitations that simulate human behavior, etc). The combinations of all these components generate a complex appraisal process that helps create believable scenarios (according to the components added).

There are some recent applications of FAtiMA architecture, like Sueca [25] (creation of a social robot player that is able to play a traditional game in a social manner) or Traveller (an agent based application for intercultural training) [26].

There is also an ongoing project called RAGE. This project goal is to boost games development for education and training in Europe. RAGE will provide a collection of self-contained gaming assets that support game studios at developing applied games more efficiently and making them better suited for their purpose. A toolkit is being made that includes a port of FAtiMA from JAVA to C# and six RAGE assets as well, that together, aim to improve and add features of FAtiMA, as follows:

- Improvements on the modular design.
- Simple integration with current game engines and other AI frameworks.
- Accessible authoring tools and proper documentation.
- Able to work on multiple application environments (Windows, Mac, browser, IOs, Android).

Some of the principles behind appraisal processes that can be added to FAtiMA are viable options to serve as basis for the decision making of our agent (the merchant), and there are several of RAGE assets may be of use as well (like the Emotional Appraisal or the Emotional Decision Making).

With that said, Rage is still in his early days and FAtiMA, even with its’ modular properties, seems to aim for behaviors and scenarios more complex than we intend to implement.

2.5.2 CiF

Comme il Faut is an artificial intelligence system and authoring strategy for creating game-based interactive stories about relationships and social interactions between characters [27]. CiF aims to reduce the burden on the author by providing a playable model of social interaction where the author provides reusable and recombination representations of social norms and interactions.

Traditional authoring methods imply an immense investment of time and resources in order to make rich and compelling social interactions and stories. [28].

CiF’s final architecture provides the following main features [29]:
• Multicharacter social exchanges are explicitly represented separately from any specific characters. Actions are chosen not based on the characters themselves, but on their traits, which are dynamic.

• Characters’ decisions are based on more flexible decision making, instead of boolean flags or rigid preconditions.

• Social interactions don’t affect just a single change in social state, they have a snowball effect across multiple characters.

• Details and outcomes of previous interactions are stored to help keeping a sense of continuity in the social exchanges.

Initially, CiF was developed as a stand alone simulation system, the architecture and authoring approaches evolved considerably with the development of “Prom Week”.

### 2.5.2.1 Prom Week

Prom week is a social simulation game about the dramatic week leading up to a high school prom. Players influence the social landscape indirectly by having the characters engage in social exchanges [4]. Prom week involves solving social puzzles, like making the class nerd date a popular girl in a set number of turns, by choosing what interactions and with whom he will make them. Its interface can be seen in figure 4, where several options correspondent to different actions are presented to the player.

We will now specify the implementations of CiF in Prom Week:

• Social Physics: This is the basis of the interactions and exchanges that can take place in Prom Week. They are composed of over 3.500 social-cultural considerations. Each consideration is a rule that can influence the characters’ desires and can have a positive or negative weight in each interaction. These rules transmit a notion of social common sense (for example: vengeful characters are more prone to have mean actions).

• Social State: Prom Week’s social state is made of six components:
  
  **Relationships** (Binary, reciprocal and public connections between characters. Prom week has: friends, dating and enemies).

  **Social Networks** (Scalar, non-reciprocal and private variables that represent feelings from one character to another. Prom week has: buddy, romance and cool).

  **Statuses** (Binary, Temporary feelings that result from interactions. Some status are private like shame while others are public, like popular).

  **Traits** (Permanent attributes of a character’s personality. These traits can be public or private).

  **Social Fact Database** (Social history of interactions, it is public and available to all).
**Cultural Knowledge Base** (The objects of the world and the relationship and/or opinion each character has of that object).

![Figure 2.4: Prom Week](image-url)

- **CiF Processes**: All of the characters’ desires and responses to interactions are determined by customizable rules encoded in CiF in order to allow, given certain scenarios, the characters actions or status to change as a result of several factors regarding the context and other characters’ in that scenario. CiF operates by looping a set of processes. Those processes are:

  **Desire formation**: Determines a volition of each character towards other characters for each social game (interactions).

  **Selection of the social game**: A social game (or Social Exchange) is selected by the player. Each has an initiator intent (what the character wants to do) and three roles (initiator, responder and a third party). Depending on the social states, social games can have different results.

  **Performance realization**: Performance realization is the set of dialog acts and animations associated with the chosen social game result. After that, a set of trigger rules are deployed to make the changes on the characters statuses and the social state. The social states and the entry are then saved in the social facts database.

In conclusion, there are some CiF and Prom Week basic characteristics that can be adapted to our work. Prom Week is a good way to describe and show of the CiF in action, but the lack of player to NPC...
interaction and the focus on multi character interactions make it so that it can’t be transposed as is, and does not reflect a proper implementation example of the work proposed by us, which focuses more on the interaction between the player and one specific kind of NPC.

2.5.2.B CiF-CK

There is a work with a lot in common with the proposed one on this paper. This work provides a social component to NPCs to the game Skyrim [9] as a mod. It takes inspiration from CiF and Prom Week in order to improve the narrative engagement and player experience for human interactions in the RPG Skyrim. CiF-CK was built as an adaptation of the structures present in CiF to RPGs. It also expands upon the original architecture by adding a Belief system and a keyword-based authoring tool. The architecture of CiF-CK is composed by the same components described before in Prom Week:

- Social-Exchanges: They are equivalent to the Social Physics detailed before, but with alterations to fit the context of an RPG better, it includes social moves like fighting, giving gifts, flirt and insult. The death of an NPC also has to be taken into consideration in this mod.

- Characters: in order to apply CiFs’ architecture to the context of a Skyrim mod, improvements to the characters were made. These improvements involve the implementation of key variables that help to capture additional behavior (those key variables fit better in the context of an RPG and with the game engine used). The Social states status and traits present in Prom Week fit into this component.

- Social State: As with Prom Week, this component represents the state of the world, and takes into account: Social Networks, Relationships, Cultural Knowledge Base and Social Facts Knowledge Base. A thing to note is that, since in RPGs nowadays, areas are loaded and unloaded depending on the player being or not in that area, the data had to be stored in a static entity so that there’s no lost of information when an area is “unloaded”.

- Trigger Rules: As opposed to Prom Week, where trigger rules can be fired at any point in the game, CiF-CK checks and fires all trigger rules when a quest ends, to be more efficient and light to the game engine.

- Beliefs and Social Networks: As mentioned before, Prom Week has three Social Networks (Buddy, Romance and Cool) [4]. CiF-CK has Attraction (similar to Romance) and Friendship (similar to Buddy), but adds an element non-existing in CiF or Prom Week, which is the application of Theory of Mind. With this feature, a new layer of depth in terms of behavior unpredictability is added,
making the interactions seem more engaging.

- **Player to NPC interaction:** Usually, in RPGs, the world is centered on the player. In CiF-CK the NPCs take the player into account in their social goals, and the player social moves (actions or dialog) can reflect direct or indirectly the perception of the NPCs. That way, the feeling of a living environment, where the NPCs interact with each other and with the player in an organic and believable way, is well achieved.

To sum it up a lot of the principle and components of this work can serve as inspiration or basis to our work, mainly, the ideas behind the Player to NPC interaction, Social Networks, Social State and some of the implementation approaches. Of course that all of this would have to be modified and hopefully improved (especially in the player to NPC interaction), in order to fit our context of a deeper interaction between the player and a specific NPC with unique characteristics (Merchant), and take those characteristics into account to provide new interactions between both.

### 2.5.3 PsychSim

PsychSim is a social simulation tool designed to explore how individuals and groups interact and how those interactions can be influenced. This tool allows a user to build social scenarios where, either groups or individuals interact with each other. Each entity of the scenario has his own goals, relationships with other entities, private beliefs and mental models about others [5].

A central aspect of the PsychSim design is that agents have fully specified decision models of others. This is due to the fact that our actions and decisions are, a lot of times, based on our prediction of the behavior of others in terms of their mental state (beliefs and desires). These predictions have been referred to as “theory of mind” [30]. With that said, these recursive decision models allow the formation and selection of a big range of factors to justify the agents choice [31].

In PsychSim, the process that allows a user to explore different tactics for dealing with a social issue and to see the consequences of those tactics is composed of three phases [5]:

- **Design Phase:** the user populates the scenario with entities and specifies, not only the models used by those entities, but also their behaviors, that would be adapted to the models themselves.
- **Execution phase:** in this phase, the user runs and controls the simulation, observing the results of the interactions between entities.
- **Analysis/Perturbation phase:** The results of the simulations are placed into a database. Inference rules analyze this database and, based on this analyses, the system reports anomalies and perturbations in the results. With this, there’s the possibility to iterate the model in order to refine it.
Agents in PsychSim are embedded with a decision-theoretic framework that allows each agent to maintain his own beliefs about the world, his own goals and his own policies to achieve these goals. The description of PsychSim can be separated into three components [31]:

- Model of the world: Each agent starts with a state (collection of objective facts that an agent has, like force or trust towards another agent), and with conditions that determine which actions translate into state changes.

- Preferences: The incentive of an agent to perform a certain action. Its based on his desires of maximizing or minimizing one or more of his or others states and/or actions.

- Beliefs about others: In order to simulate human behavior (and not just apply a maximization algorithm that does not reflect human error and misconceptions), beliefs about others consist of Nested Beliefs (The agent subjective view of the world, it can have several levels since an agent can have a belief about what another agent’s beliefs), Policies of Behavior (represents how an action is chosen based on the agents beliefs) and Stereotypical Mental Models (templates of mental mod-
els that agglomerate a set of specific beliefs accordingly to the mental model it wants to represent).

2.5.3.A SocRATES

SocRATES is a computational model of social attitudes with the purpose of building a virtual agent able to reason about its social role and its social relation towards the entity he is interacting with, and thus select its actions accordingly [6].

In the context of this model, PsychSim is used for defining a set of actions and their influence on the state of the world.

The social decision model that SocRATES uses has the following components that takes into consideration when defining the agent mental state [32]

- Social role and social relation: Defines the agents function during the interaction and the ideal state of the relationship for the agent (according to the agent he is interacting with) given the situation.
- Long-term relational goal: Describes the importance granted by the agent to maintain a long-term relation with another agent or user.
- Social goals: Related to the “Theory of Mind” and represent how an agent would like to be perceived by another agent.
- Situational goals: Are defined by how the agent considers its own social role.

![Figure 2.6: SocRATES Global Model](image)
2.5.3.B Thespian:

Thespian is a framework for realizing interactive drama that seeks to turn authoring from a programming burden into a creative exercise. To accomplish this, Thespian “train” the agents of the interactive drama to perform their roles.

This framework allows the author to create alternative scripts of the desired paths of the story and specify the story environment (how the agents’ actions affect the state of the world and the beliefs of themselves and of others).

With this, there is no need to script every interaction, the author simply has to add constraints on agents’ behaviors [33].

The PsychSim usage as basis of this work brings several advantages: the support of multiple interacting agents with their individual goals, the subjective views that they have of the world, themselves and of others, and the capacity of translating desired behavior into the goals required to perform said behavior.

2.5.3.C BiLAT:

BiLAT is a game-based simulation and tutoring system developed to provide students with an environment to train and simulate bilateral negotiations.

The behavior of the characters in this simulation is separated into two models [7]:

- Dialog Model: responsible for governing the turn-by-turn dialog.
- Negotiation Model: responsible for driving the formal negotiation of offers and counter-offers.

The negotiation model uses PsychSim mainly because of its capacity to model entire scenarios with diverse entities that interact among themselves. It also takes advantage of the fact that each entity in PsychSim has individual beliefs and mental models of others.

2.5.4 The Friendly Blacksmith

Similar to this paper, there was another work that tried to address some of the problems regarding the interaction between the player and a merchant NPC. This work is called “The Friendly Blacksmith” and its goals are to make it easier for the player to find the items he wants in the shop. This is done by presenting to the player a list of items that the merchant deemed to be the more probable for the player to be looking for.

This was done by creating a recommender system. The friendly blacksmith takes into account information about the player in order to order and recommend items for him [34]. The model itself is made of five different components:

- Main Module: Stores the preference model and communicates with the game itself.
• Minimax Regret Module: Decides which item is most appropriate for a recommendation.

• Elicitation Strategy: Requests more information from the player when necessary.

• Current Solution Strategy: Decides what to ask to the player in order to maximize.

• Recommendation Strategy Module: Responsible for constructing a list of recommendations based in the item selected in the Minimax Regret Module.

Results of tests with and without this model revealed that, when using this model, the game experience was more enjoyable, recommendations were more appropriate, more recommended weapons were bought and it was noticeable how queries had a more positive effect on subsequent recommendations.

The main concept of this model, which is to improve the player experience when interacting with merchants, matches our work’s objective. As such, we will consider implementing a very simplified version of it, because our focus is more towards social interactions. On the solution architecture, a more detailed description of the ideas taken from this work will be explained.

It is interesting to think of a project that could merge the friendly blacksmith with our work. This could take advantage of the strengths of both thesis, providing a merchant that would have a good and believable social component and kept the shop easy and intuitive for the player to navigate in.
2.5.5 Misrepresentation Game

The “misrepresentation game” is a game that focuses on the formalization of the notion that people can, and might, lie about their own interests in order to maximally benefit themselves while creating an illusion of fairness towards the other party.

Analysis and results using several negotiation strategies show that many of them can yield strategic gain. That gain can be achieved even when the opponents’ preferences are unknown from the start (by making a trade of information, where the information offered can be untruthful) and that people find these strategies credible and even preferred to honesty. [8]

In summary, agents can win negotiations by lying while seeming fair. The best strategy studied to apply this is a type of lie called “fixed-pie lie” that involves the agent pretending to have the same preferences as their opponent (as shown in Figure 8).

Unfortunately, it's not our focus to implement a negotiation system in our work, but it is a very interesting possible addition, and this study can be very useful in that situation.
2.6 Games

In this subsection we will analyze different video-games that can serve as a basis to pinpoint some features and limitations of merchants and shopkeepers in this medium.

In all these games, the interactions with shopkeepers are essential not only to keep the flow of the game going (buying more advanced items, items necessary to progress in the game or even because they are a key part in some quest) but also because they help creating a more rich and alive world.

But, as we will see, there are still some limitations about those merchants that break the immersion of the player in the game, mainly because their interactions are unnatural for a normal person to make. Sometimes these limitations are applied because it makes the interactions faster and, don’t forcing the player to spend much time there, because it’s not the main focus of the game. But hopefully there’s a solution that can accommodate both the immersion and practicality of interactions with merchants.

2.6.1 Witcher 3

The Witcher 3: Wild Hunt is a open world RPG where the player goes from place to place in a rich and live environment to complete quests and missions. A big part of the game is played in cities and villages where the player can interact with all kind of NPCs, including merchants.

![Normal merchant interaction in Witcher 3](image)

Figure 2.9: Normal merchant interaction in Witcher 3

These merchants have some positive features and details that contribute for a better representation of reality like the fact that the money that each merchant offers for a certain item, depends on his specialization (for instance, blacksmiths will offer you more money for a sword than other merchants) and that each merchant has different amounts of money to buy the player’s items (the money they have is reflective of their city’s economic status).
Unfortunately, a lot of the usual problems of other RPGs when it comes to NPC behavior still prevails, more noticeably the fact that player actions towards the merchant are very limited (there are usually only two or three dialog options, and neither of those have an impact on future interactions with the merchant). There are some exceptions of merchants that can change behavior depending on the player choices in some quest related to them, but these interactions are still very limited, they come through as something predetermined and it is clear that a lot of work was put into the authoring of them.

2.6.2 Animal Crossing

The Animal Crossing series are community simulation games that revolves around the life of the player in a new and calm village. In this village the player can talk and interact with their inhabitants and perform all sorts of tasks like fishing, bug catching etc.

The merchant of this village is a very important NPC to the series, being both iconic and relevant to the progression of the story. The player has the option of engaging on a wide variety of interactions with this merchant. This character has some very unique and interesting features, rarely seen in video-games, that help his shop to have real live characteristics that are lacking in other games. These features include the fact that the game uses the internal clock to have seasonal items for sale (like Christmas items in December) and to sell raffles in the last day of each month.

Another interesting feature is the evolution of the shop itself according to how much money the player spends there. The shop will grow bigger and sell more items once the money spent by the player there reaches certain thresholds.

These are all interesting ideas that give more depth to a single shop and merchant, but nonetheless, the range of interactions is still very limited and there is no real relationship between him and the player.

It is still worth noticing that other RPGs can gain with the implementation of simple features like these ones.

2.6.3 The Merchant Life

Some games focus entirely on the merchant and the economy of the game, and, as such, a lot of effort is put into improving and refining this part of the game. But these games put the player in the place of the merchant, this is, it is clear that the investment made to make the trading component of the game as complete as possible only came to be because it is the center focus of the game itself.

Nonetheless, these games can be viewed has a source of inspiration about features to be implemented, for example, in the game “This Merchant Life” [35], the player has control over a merchant in the medieval era, and it has at his disposal mechanics such as:

- A reputation system that grows through funding projects and constructions within each settlement he passes through.
• Opportunity to grow as a merchant by investing in such things like a better horse to carry your cart.

• A prices are dynamically attributed, depending on the settlement the player is doing business in

This game obviously lacks the social component we want to give to the merchant, because in this case, the merchant is not an NPC but the player itself. But, it is still important to learn and record some ideas and features that exist, because the model proposed in this paper can probably be improved by them.
2.7 Dialogue Systems in Games

Dialogue is one of the main forms of communication, its importance in unquestionable. In videogames, instances of dialogue can be seen across all genres, because it is a good way to convey information regarding the story of the game, the personality of a character, or any other data.

Uses of dialogue can range from really simple ones, like a fellow soldier pointing out where the enemy troops are in a shooter, or complex ones like the intricacies of a dialogue tree in an RPG.

There are several types of dialogue systems possible to be implemented in video-games, but we will focus on two of them:

- **Non-Branching Dialogue**: This is the simplest form of interaction related to dialogue. It involves a non-interactive action by the NPC when something triggers. When the NPC delivers its lines of dialogue, the conversation ends, there is no control over the conversation for the player (for example in Fig.2.12).

  ![Figure 2.12: Legend of Zelda](image)

This type of interaction is very common due to its simplicity and can be the best option in certain situations where the information to be passed is not very relevant or requires a lot of focus from the player.

- **Branching Dialogue**: Video-Game dialogue becomes more interactive when conversations can take different paths depending on the player choices [36]. Branching dialogues are conversations where the player can choose what to say, in order to advance in that conversation. This implies that the amount of authoring work is greatly increased due to the fact that a lot of new dialogue
lines have to be written and it is not guaranteed that the player will experience all of them. There are techniques that minimize the authoring effort, for example having the illusion of freedom, by giving the player a choice that sooner or latter, lead to the same outcome [36].

Despite of this, it is clear that branching dialogues improve the immersion of the player by giving his choices weight in the narrative of the interaction.

The work done in this paper ended up using both techniques depending on the situation.

2.8 Modding

Computer game modification, or "modding", is an important part of gaming culture as well as an increasingly important source of value for the games industry [37]. There are a number of successful games that are the result of modifications of other games, it's the case of Defense of the Ancients [38] that it's a mod of the popular game Warcraft 3 [39] or Counter-Strike [40], that was originally a mod of Half-Life [41].

However, not every mod translates into a new game, the majority of mods are just modifications or improvements to the original game.

![Figure 2.13: Mammoth chickens mod: Adds giant chickens to the video-game Skyrim](image)

Some games provide software to allow the players to mod, the game The Elder Scrolls: Skyrim [9] and other Bethesda games have a tool called Creation Kit that allow the modding of those games. Another, more recent game that have a modding tool available to the player is Conan Exiles [42] that has the Conan Exiles DevKit, a custom build of Unreal Engine that allows the player to modify a great part of the game assets.
2.8.1 Conan Exiles and DevKit

Conan Exiles [42], is an open-world survival RPG, planned to be released in 2018. This means that, at the time of this work, the game is in early access, a state where the game is playable but still in constant alterations and updates.

In Conan Exiles, the player takes control of an exile and has the objective to survive in the harsh environments, build a base and dominate over the other players. The game is a massive multi-player online game, which means that several players can interact with each other in this world.

As previously stated, Conan Exiles has a modding tool called DevKit 2.14. This tool uses an altered instance of the Unreal Engine in order to allow the player to have control over models, sound files, A.I, maps and other components of the game.

![Figure 2.14: Conan Exiles DevKit launcher](image)

There are limitations though, the DevKit doesn’t allow changes or searches on some base classes defined in C++, the user can only call them. Every class that the user can change, is defined in blueprints visual scripting (a gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor).

The important concepts to grasp about the DevKit in order to understand our use of it are:

- **Data Table**: DataTables are tables of miscellaneous but related data, grouped in a meaningful and useful way.

- **Blueprint**: Blueprint is an asset that allows content creators to easily add functionality on top of existing game play classes

- **Actor**: Actor is any object that can be placed into a level.
- **Event**: Executable method that has no return value and do not have any output parameters.

- **Function**: Executable method that has at least one output parameter or return a value.

Another characteristic not directly related to the DevKit but still important to mention is the fact that there is a strong community surrounding mods in Conan Exiles that more times than not, are glad to help. Some developers of the game itself are very accessible as well in terms of communication. There are very active dedicated channels to discuss mods in Conan Exiles.

### 2.8.1.A Problems and limitations

At the time of this work, there are a considerable number of problems in the Conan Exiles DevKit. Fortunately, most of them are a result of the game being in early access, so it is to expect that those problems will be solved in the future.

One of the major problems is the fact that, due to the game being in early access, there were updates to both the game and the devKit very frequently. These updates could potentially make the mod crash, because either the update affected a blueprint that the user had change, or the update affected a blueprint that had dependencies with something that the player changed.

As previously stated, not every part of the game is modifiable, there are classes defined in C++ that users have no access to. We can call functions and events of those classes, but we cannot access them or change them. This may not be a problem in a lot of mods, but when the user sees a call to an event and that event is defined in a C++ only class, then the only option is for the user to assume how the function works and what it does.

Another problem came from the lack of documentation regarding the game. Currently, a Wiki is in development to help get started in the DevKit but it is in an early stage (at the time of this work). This is particularly serious because there are classes and functions that are not accessible to the user, as such, when calling such a function, without documentation, there are no way of knowing what that function does.

These three main step-backs cause a lot of other problems, like the fact that only some of the classes can make calls server-side or that the ID of a character can’t be compared using a normal method. We will talk about both of these problems in particular during the implementation chapter of this work, but this goes to show that a lot of problems, due to a lack of documentation can be hard to solve.

Despite all this, Funcom is working and constantly updating the game and the documentation, this means that it is important to emphasize that these problems were present during the development of this work, because they may be solved in the future.

---

1 The developer of Conan Exiles
2.9 Discussion

All of the fields of expertise mentioned above can be of use when developing our model.

From the games mentioned, we can take the good ideas they implement and try to improve them, while trying to solve some of the limitations found. For example, the fact that merchants in Witcher 3 (section 2.6.1) show preference for buying items similar to those that they sell, is a feature that makes sense and helps with the immersion of the interaction. Our proposed model will cover that feature while preventing some unrealistic behaviors like the fact that any kind of aggressive actions have little to no long term consequences.

The psychological models and studies about merchants and sales techniques will be taken into consideration in order to allow the player to have believable interactions with the merchants in game. For example, the Friendship Model (section 2.4.2) will help to delimitate stages of the relationship between the merchant and the player. While the sales techniques (section 2.1) can be implemented by the merchant itself depending on a variety of conditions. By making the merchants have more realistic actions and behaviors, we are contributing to their believability.

The state-of-the-art architectures and models will be the basis of the implementation techniques used. By relying on those works, we can more easily create the proposed model. In particular, a lot of the model will be based of CiF-CK (like some components and part of the interactions between them) since it is the one that most resembles our proposed work. A detailed description of our model will be given in the next section, where the similarities and differences between our work and CiF-CK can be seen.
3

Solution Architecture: A Merchant Model

Contents

3.1 Overview ................................................................. 33
3.2 Information Flow ....................................................... 35
3.3 A Dialogue System Model ............................................ 36
In this chapter we propose a merchant model to be used in RPGs that aims to fulfill the goals established in the chapter 1. We will go into detail about the proposed architecture, from where it took inspiration and which adaptations we did in order to contextualize and innovate this model.

There is a lot room for improvement when it comes to NPCs as stated in the Introduction chapter. This work in particular will focus on improving the interactions with merchant NPCs in RPGs with the purpose of making them feel more believable and, as a consequence, improve the experience of the player.

The model created aims to give the merchant a psychological as well as a strategic dimension. These dimensions will make the merchant not only adopt a more realistic approach as a businessman (by wanting to profit and by making several different kinds of deals), but also make it feel more human (by having the possibility to develop a relationship with the player).

The relationship between the player and the merchant can be developed in different ways and at different paces, because it depends on a variety of factors. Not only that but also the possible outcomes of said relationship may also vary depending on the factors mentioned.

The merchant model designed, takes inspiration from other architectures, more specifically the principles adopted by CiF and CiF-CK for example: the use of some of the components Social State like the Relationships and Traits. The model also looks at Fearnout AffecTive Mind Architecture (FAiMA) use of Memory as a relevant factor for future actions.

When developing the model, it was important to make it generic enough to be easily implemented in different RPGs and modular enough to be expanded or simplified according to the needs of the developers.

A more detailed description of the model will be given in the next section, but a overview can be seen in the Fig.3.1.
3.1 Overview

In this section we take a look into each component of the model proposed, describe them and their functionality and finally, see a step-by-step information flow of this model from the point where the player interact with the merchant, until the output is decided and presented.

3.1.1 Player Action

The player action is an event, not a component, it is responsible for starting the model iteration and can be anything deemed justifiable, in the case of a merchant for example, player actions can be dialogue options, the act of buying something, the act of offering something, or even a less obvious action like offering items or completing a request for the merchant.
Each action of the player brings with it a value that represents the weight of that action. That value will affect considerably the result of the interaction with the merchant. Every weight can be personalized, according to the importance of the action, for example: buying a cheap item can be less relevant than offering a large amount of items to the merchant.

3.1.2 Memory

This component of the merchant is made of dynamic information and is separated in three different items:

- The social fact database: Responsible for saving the past interactions between the player and the merchant.
- Social networks: Variables that correspond to how the merchant sees the player in terms of: Monetary Interest, Perception and Social Bond. Where the Perception can be seen has the opinion of the merchant based on what he knows of the player and the Social Bond as the the opinion that the merchant has of the player based on past interactions and the growth of the relationship between them.
- Player preference database: This is basically a list of items or actions that the merchant knows the player likes, for example, trough dialogue questions, the merchant may infer that the player likes swords more than shields. That information is saved in the player preference database.

3.1.3 Cultural Knowledge Database

This component corresponds to the list of objects or actions that the merchant values more or less, in short, it’s a database of his preferences. This makes it so that, when interacting with him, he can value more some actions in relation with others, because he finds them better. For example, a merchant that really likes herbs, can value more the fact that the player offered him herbs instead of a sword.

3.1.4 Traits

Traits are fixed properties, unique to each merchant. They are the personality characteristics of the merchant and can affect the outcome in different ways. The traits can be divided into two types:

- General traits: Friendliness and grumpiness are examples of these traits. They affect the social networks values, making them weight on the decision of the relationship status.
- Merchant specific traits: Traits like greediness or the ability to make good deals, these affect more directly the decision making, not contributing for the alteration of the relationship but instead, affecting the prices the merchant is willing to accept as payment for different goals (for example).
3.1.5 Relationship

The Relationship component is responsible for keeping or updating the state of the relationship between the merchant and the player. The relationship is updated according to alterations in the social network values. The states are inspired on the ones explained in the friendship model (from the section 2.4.2).

The state of this component is responsible for affecting directly the decision making. For example, if the player is in the state of "Close Friends" and drops to the "Casual Friends", the merchant may remove the possibility of the player to buy a certain item.

States may be reached when a certain condition dependent on the network values is found, for example, to reach the "Casual Friends" state, the player might have to decrease substantially the values of monetary interest and social bond.

3.1.6 Decision Making

This is the last component of the model and it is responsible for determining, depending on the relationship state, the player preference database and the merchant specific traits, which will be the outcome interaction. This outcome can come in many forms, for example:

- A change in the shop content (for example the merchant can make available an item that was unavailable prior to this interaction)
- A sale or inflation of prices in the shop.
- A quest can be triggered.
- A simple dialogue option is played.

The criteria itself for each outcome decision is up to the developer to make. The objective is to provide the tools to easily determine an outcome and the criteria for that outcome to be triggered. This would hopefully cut on the authoring effort.

3.2 Information Flow

We will now describe an example of an iteration of our proposed model, where the player sells a sword to the merchant:

- The player realizes the action “Sell Sword”.
- This action is saved in the Social fact Database and is taken into consideration in the Social Networks.
• Cultural Knowledge database is going to identify a “sword” as an item that the merchant likes and it will increase the value of that action.

• In this case, the Player Preference Database will not suffer alterations since the action of the player does not reflect any preference he has about some item.

• The “friendly” and “grateful” traits that the merchant possesses will also inflate the value of the action because they support the “sell” action.

• The values added by the traits and Cultural Knowledge Database are applied to the Perception component and the Monetary Interest, while the Social Fact Database will evaluate this action and previous actions in order to add a value to the social bond component (in this example we assume that previous interactions were positive to the merchant, so the social bond value is increased).

• The new values of the Social Network are passed to the relationship component and those values will determine in which state of the relationship the player is. Let’s say that the player goes from the Acquaintances state to the Casual Friends State, given the updated Social Networks values.

• Now, in relationship status, the merchant specific traits and the player preferences database will go into the decision making component which will choose a strategy to apply and/or a dialog option. The strategy will most likely end up updating the inventory with new items and prices.

A visualization of the information flow described above can be seen in Fig. 3.2.

3.3 A Dialogue System Model

The proposed merchant model presupposes the existence of a relatively advanced dialog system. While most RPGs already possess, Conan Exiles, being still in development as of the time of this work, lacks such a system. Here we present our dialogue system model, made to solve this problem.

Our solution for making a viable dialogue system had to guarantee that there were enough lines of dialogue to make the interactions with the merchant feel non-repetitive and that the lines reflected accurately not only the current interaction with the merchant, but also the relationship status between the player and him.

As such, we broke the dialogue system in two main types of dialogues, the ones that were not related to dialogue lines chosen by the player (we will treat them as reaction dialogues), and the ones that were.

3.3.1 Reaction Dialogues

To make the reaction dialogues, we set up a enumerator of possible actions (non-dialogue ones) done by the player, that justified an reaction from the merchant. These dialogues can be considered
non-branching dialogues (explained in section 2.7) due to the fact that are triggered from an action but there is no direct interaction that the player can have with it.

**Table 3.1:** Actions that justified a dialogue line from the merchant

<table>
<thead>
<tr>
<th>Action</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greetings</td>
<td>When interacting with the merchant</td>
</tr>
<tr>
<td>Farewell</td>
<td>When stop interacting with the merchant</td>
</tr>
<tr>
<td>Ask Item</td>
<td>When interacting with the merchant after he asked for an item but before the player delivered said item</td>
</tr>
<tr>
<td>Open Shop</td>
<td>When asking the merchant to open the shop</td>
</tr>
</tbody>
</table>

In our implementation, we gave each one of the actions (described in the table 4.1) six possible dialogue lines: three for when the player had a positive relationship with the merchant, and three for when the player had a negative relationship with the merchant (the definitions of positive and negative relationship will be explained further ahead).

It is relevant to point out that we had two instances where the merchant asked for an item (we will explain this later on), so the Ask Item action had double the dialogue lines, six for each item.
Ideally, this should be less binary and more fluid, and despite being built in a way where the number of actions and conditions to say each lines can be fully customizable, due to time constraints and the fact that the dialogue system was not the top priority in this work, we had to move our resources to other areas. It is important to point out that this system can be expanded to meet with each developer’s necessities.

3.3.2 Dialogue Tree

To make all the other dialogues (the ones that are part of a conversation, this means, branching dialogues) we used a system of five binary trees with height four. The nodes on each tree (Fig. 3.4) consist in an alternation between the player’s chosen dialogue and the correspondent merchant response. After the player reaches the last node of a tree, the following tree is chosen according to the relationship status between the player and the merchant. If it is positive, then the player goes to the next positive tree, otherwise it goes to the next negative tree (as seen in Fig. 3.3).

![Figure 3.3: State machine of the dialogue tree system, where the criteria of the tree selection is the relationship value being above or below 5 (this value is customizable)](image)

The last two trees (positive and negative) loop between each other or themselves, so, to try to avoid repetition of dialogue too much. Each player interaction has three possible (and identical in meaning) merchant answers, so that if the player goes to the same tree two times in a row, it doesn’t feel too
In order to implement this system, we started by using a software called Yarn to make and organize the trees. This let us have an easier idea of how the interactions were going to feel, because by having the tree mapped out, we could easily follow an interaction to see if it made sense.

Unfortunately, the fact that the tree is binary, it means that the choices end up to be a bit binary as well. For our system, we used the same weight for all positive options and the same for all the negative ones. This isn’t ideal, but given the circumstances, it works fairly well. It would be better if the tree was broader in order to fit in more neutral options. Despite the broadness of the trees being fairly complicated to change given the method we use to develop this system, the weight of each option (positive or negative) can be changed with little effort, giving more flexibility in the authoring process.

Another feature we added was to integrate small quests in order to go from the first tree to the second and from the second to the third. The quests consisted on the merchant mentioning that he wanted a certain item, and the player had to offer that item to the merchant to be able to continue talking to him. We did this because it incentives exploration, it makes the interactions with the merchant last longer, and to show how small quests can be integrated with the dialogue system and can work as extra good or bad actions in the model we created. Despite being relatively simple, it proves that there is room to seamlessly integrate another type of interaction with the dialogue.

The criteria of the selected tree and a detailed explanation of the inner works of the dialogue tree will be explained later when talking about the specifics about the implementation.

Figure 3.4: Dialogue tree used on a merchant
3.3.3 Summary

The amount of authoring of this system, for the most part, is flexible to meet each utilization requirements. It has a lot of room to be improved, but once again, it wasn't the main focus of the thesis, only a tool made to fit the requirements of the model we though about (a full view of the dialogue trees existent in a merchant can be seen i Fig.3.5).

Figure 3.5: Full set of dialogue trees existent in a merchant
4

Implementation

Contents

4.1 Implemented Model ................................................. 42
4.2 Shop .............................................................. 44
4.3 User Interface ...................................................... 46
4.4 Implementation Details ............................................ 49
In this chapter we will focus on the model we chose to implement, on the implementation environment and process as well. As we will see, the model implemented was based on the proposed model (see chapter 3) but compromises had to be made due to reasons we will specify in depth in this chapter.

We will start by explaining why did we ended up abbreviating the proposed model explained in the previous chapter, then we will see how the model used works, we will see how the NPCs and shops were created, how the dialogue system works and then we will go in depth on some implementation choices, problems and solutions.

### 4.1 Implemented Model

The model implemented is an abbreviation of the proposed one, this happens mainly because there were a number of setbacks while working on the Conan Exiles DevKit. The main one is the fact that we are working on live code, from a game that is in early access, meaning that the number and frequency of changes made to the code is very high. This means that a lot of the work done can be overwritten by an update to the devKit (that, by the time we were implementing the core of the model, happened almost on a weekly basis).

Due to this, cut the maximum dependencies possible to classes that are from the game (because if those classes were updated, all the code done in them could be overwritten and the dependencies may stop working).

Another limitation of working on the Conan Exiles DevKit is that, the users don’t have full access of the game code, only classes determined by the developers, presented in blueprints, all parts of the game defined in code are un-accessible for DevKit users. We can call functions that are defined in a C++ script, but we have no access to it’s definitions, that can make it hard to interpret some functions available base on the name and arguments only.

On top of this there were more specific problems like the fact that we had to pivot the work to a creation of a dialogue system (that was not the focus of the thesis) and the creations of merchants with working shops.

Despite this, the model we implemented does not compromise our objectives because we focused on creating the most relevant components and setting up a test environment that allow us to get the information we look for. The work tested can be seen as a base model for the proposed one, everything added it’s an expansion.
4.1.1 Components

The components created for the implemented model have the same functionalities of the ones specified in the proposed model, but with some differences due to the limitations stated before and the engine we are working on.

- **Interpreter:** This component takes into account several factors in order to update the relationship status between the player and the merchant. After updating the relationship status, it also updates the Social Fact Database.

- **Traits:** These are fixed values that represent traits of the merchant. These values determine what is the personality of the merchant as well as some of his characteristics. There are two types of traits in this model.
  - General: Traits that correspond to characteristics that the NPC has as a person, for example, nice, rude, arrogant, etc. These traits will influence the interpretation of the player action.
  - Merchant: These traits are specific to merchants, which means that they are related to characteristics that are present in business people. They can be, for example: Greediness, ability to make business, con-artist. Merchants’ traits affect directly the Decision Making component.

- **Memory:** This component is comprised of two others, the Social Fact Database, and the relationship status and it represents data that can be changed in runtime:
  - Social Fact Database: It’s is the memory of past interactions, it saves the relationship status after each interaction in order to take that into account when interpreting an interaction and deciding the new relationship status.
  - Relationship Status: It is the value that translates into the relationship status, in this implementation, the value ranges between 0 and 10 being 0 the worst possible relationship and 10 being the best possible relationship.

- **Decision Maker:** This component is responsible for deciding an outcome based on the current relationship status and the merchant specific traits. The decisions the merchant we implemented can take and the criteria for those same decisions will be specified in the subsection 4.1.2.

4.1.2 Information flow of the model

A normal iteration of this model can be described as such (Fig.4.1). We can see the simplifications done in relation from the model presented in Chapter 3. In here, the information is more centralized and we don’t have implemented the preferences of both the merchant and the player.

1. The player performs an action that affects the merchant, for example, saying something positive.
2. The Social Fact Database will provide the interpreter with the past interactions between the player and the merchant.
3. The interpreter will also get the current relationship status and the general traits of the merchant.
4. Now, the interpreter will take into account all the data received in order to determine the new relationship value.

5. After getting the new relationship value, the social fact database and the relationship status component are going to be updated.

6. The decision maker component will now take into account the new relationship status and the merchant specific traits in order to calculate the outcome of the interaction.

![Implemented model](image)

**Figure 4.1:** Implemented model

### 4.2 Shop

In this section, we will discuss what shops the merchants have and how they work. There are a total of three shops implemented in this project, one of them being for tests purposes only (as we will see in the the section 4.5.4 and Chapter 5). The reason for the utilization of 3 different shops is explained further ahead in the sub-chapter 4.5.4, but, basically, one is the shop itself (that has features such as dynamic inventory and prices), the other is used only to offer items to the merchant, and the last one is a base shop without any of the features stated above. In the following sections you can see the details of each shop:

#### 4.2.1 Dynamic shop

This is the main shop of the project and it is affected by the outcome of the model described in the chapter 4.1. When the player first opens up the shop, the state of that shop can be described as the "base state", on top of which alterations in prices and in the content of the inventory can occur. These
alteration are triggered by the outcome of the model. The features we have implemented and the way they are triggered are:

4.2.1.A Change in price

The prices of an item can increase or decrease according to the relationship status. These changes are done by adding or removing a percentage of the original cost. The way that this is implemented is as follows: after a change in the relationship status, the shop sees if that value is within a certain interval, depending on that interval, an increase or decrease can happen. The variation of the prices themselves are dependent on a merchant specific trait called “Greediness”, this allows the developer to create different merchants that change their shop differently just by changing the value of the “Greediness” trait.

A merchant with a low value of “Greediness” (not greedy), will decrease more the prices when they should be decreased (when the relationship value is on a certain positive interval) and a merchant with a high value of “Greediness” (greedy), will increase prices above their normal value (price inflation) when the relationship value is within a negative value.

4.2.1.B Change in inventory:

As for the prices, the inventory also changes according to the relationship status between the player and the merchant, but those changes don’t depend in anyway on the merchant traits. As it is implemented, the shop can put or take in its inventory 2 additional items. These items are added or removed according to certain intervals. For example: if the relationship value is above \( \alpha \), the item A is unlocked (if it is below, the item is locked).

4.2.2 Offers

In the model we designed, there are several actions that the player can take in order to interact with the merchant. One of those actions implemented was the offer option. The most obvious solution when it came to implementing this feature was to make a parallel shop, where the items in the inventory weren’t for sale, but instead, the player’s items available for offering were displayed. This means that the player could see the item that he can offer to the merchant in the shop inventory. This particular “shop”, has the same feature, that allows adding and removing items from the inventory (which means that the player can offer more or less items to the merchant) but instead of those additions being dependent of the relationship value, they depend on the dialogue tree.

When interacting with the merchant, the player can come to a node in the dialogue tree where the former says that he is looking for a certain item. Then, the option to offer said item is unlocked and the
player can make the offer in order to proceed in the dialogue tree. This makes it so the offers work more like a "quest" type of action. Nonetheless, the system can be changed to work in different ways, like for example, having the player offer several items at any time. These actions can ultimately affect the relationship status between merchant and player.

Yet, this was not implemented this way because, we need a way to avoid a situation where the player could repeatedly click on dialogue options and finish the dialogue tree within seconds. Locking the dialogue options behind "quests" is a way for the player to vary the actions he has to make in order to advance in the relationship between them. It is important to mention that the conditions, items and outcomes of the offering system can be changed to meet each situation properly.

4.3 User Interface

The user interface of this work has two main use cases, one where the player is talking with the merchant, and the one that represents the shop of the merchant.

In Conan Exiles the NPCs that had a dialogue component attached to them were basically just a normal, non-aggressive NPC that, when the player interacted with them, would say a greetings line, and open a UI box that had a button labeled "Talk". Upon pressing that button, the NPC would say one line taken from an array of lines. The NPC would say the lines of the array one by one each time the player pressed the "Talk" button, and would start the array over when it was finished.

Some of the lines that were present in said arrays, were answers to questions the player didn’t ask (because the button only says "Talk"). This could affect negatively our work, because all of the immersion that could be caused by the dynamic shop would be loss if the lines said by the merchant were always the same and didn’t work as a reaction to the player interaction.

4.3.1 Dialogue UI:

What we did for the first use case was to use the existing dialogue UI but changing the content and amount of buttons present in that UI. When interacting with a merchant there are two possible UI states the player can find: either the player can see the dialogue options and the shop button 4.2, or the player can’t see the dialogue buttons but can, instead, see an offer button that opens the offer menu 4.3 (the reason for the dialogue buttons being locked is explained in the sub-section 4.3.2).

The update on the UI is done upon an iteration of the model, this allows for the UI to be instantly updated when the player chooses a dialogue option or offers an item, because both actions trigger the model.
In total, for each merchant, we wrote 70 dialogue lines (for the player dialogue lines in the dialogue tree) and recorded 156 dialogue lines (from which 126 are from the dialogue tree and 30 are from reaction dialogues).
4.3.2 Shop UI:

The shop UI was taken from the original game's crafting station UI, with some changes in behavior to be more representative of a shop where you can make transactions and not so much of a station where you can produce items from materials. To do this, firstly we removed the requirement which forced players to gather a list of several materials in order to "buy" an item. Instead, we now can buy this same item by using just one of the previously required materials, making it so the player can choose the payment method for a certain item.

We could create a new item that could serve exclusively as currency (like a coin), but, given the context of the game (being about exiles in a barbaric time), we thought it would fit better if purchasing was done solely by trading instead of currency.

We changed the labels on the buttons to match the purposes of a shop instead of a crafting station. Unfortunately, as we will specify ahead, it was impossible for us to change some of the UI components of the shop in order to match the current discounted or inflated price, however, we solved this by showing in more visible letters, the current price of an item and kept the old price to serve as base of comparison.

Creating the offer UI was a very similar process as the shop, though the discounts/inflation component was removed and the buttons relabeled to match their purpose.

Figure 4.4: UI of the shop with instructions for the testers
4.4 Implementation Details

In this section we will talk about the details of the implementation of the model in the Conan Exiles DevKit. This, along with the other sections of this chapter and the related work about the DevKit should give a clear view of the inner works of this model.

A lot of what we use to make the merchant and his interactions work, are re-utilization and adaptation of classes and code from other areas of the game, not only because it saves some time, but also because of the fact that due to some of the limitations of the DevKit, it’s simpler to adapt something we know it is already working.

We can see a detailed information flow of this works implementation in the A.

4.4.1 Structures and Enumerators

The more relevant structures and enumerators created are responsible for organizing the dialogue tree system. With that said, there are 2 enumerators that deal with this: The Tree Names’ enumerator (that gives a name to each possible Tree) and the Nodes’ Enumerator (that gives a name to each node of a Tree). By having these enumerators, it becomes simpler to create structures that make it easier to organize and access certain Trees and nodes. For example, the Player Option Structure has 5 Tree Names members (one for each tree) and each one of that member has all the possible nodes (taken from the Nodes Enumerator). Each node, in turn, has an array of strings correspondent to the dialogue the player can choose (it is an array because there is an option to put several similar dialogue text attached to the same node, to avoid repetition).

![Relationship Structure](image)

**Figure 4.5: Relationship Structure**
Another essential structure created is the player relationship structure. This structure contains all the essential information of a player regarding his relationship with the merchant. It contains the history of relationship values, the options chosen until now, the ID of the player and more (as can be seen in the image 4.5).

### 4.4.2 Tables

DataTables are used to store relevant information, they are easy to access and help keep all the files and information (line dialogue lines or merchant traits) organized and centralized. There are some limitations with the DataTables though, the main one being the fact that they are non editable in run-time. This makes it so that it is not easy to add content to, for example, a shop or to change values of merchant traits. There are workarounds to this problem though (as explained in the section 4.3.1).

Another problem that existed for some time was the fact that, any update in the DevKit that would update a DataTable already changed by us, would override the old table with the new one, and we would lose all changes. Fortunately that problem was solved by the introduction of merging tables. With this, we can make our DataTables independent from the ones belonging to the main game and then we can merge them. This way, we avoid having our table be overridden.

The list of DataTables with a short description of each one can be seen in the A.

![Merchant Traits DataTable](image)

**Figure 4.6:** Merchant Traits DataTable
4.4.3 Merchant

To make the merchant itself (the actor) we copied a class that represented all the dialogue NPCs (BPHumanoidNPC) and worked on top of that copy to add everything we wanted. The class itself only had an event for when the player interacted with this actor (the dialogue NPC) and had a dialogue component attached (with the simple behavior described in the dialogue section of this chapter).

Given this, we knew we wanted all of what the class had to offer but we also wanted to expand in terms of interactions and features. So, our new class (called "Dialogue NPC Merchant") also has the event for when the player interacted with him, and it also has a dialogue component attached, but we added the creation of a base shop, a normal shop and an offer interface (all these three terms will be described further in this section, under the subsection “Shops”). On top of that, we attached our own enhanced version of the dialogue component (that we will describe on the "dialogue component" subsection).

This class also serves as a middleman for calling events that are to be run server-side. The reason for this is because, by the time the multi-player component of this work was starting to be made, the only classes that could call functions server-side were classes that inherit from a specific class called ‘FunCombat PlayerController’. So, as we will see further ahead, I created a component that belonged to ‘FunCombat PlayerController” and did all the calls to events to be run server-side there. Those events were all in the “Dialogue NPC Merchant” (for organization and simplification reasons) and they would call the correct functions.

4.4.4 Dialogue Component

This component was arguably the one with most alterations done to it. The original Dialogue Component had simple verifications (like when to open up the UI and when to give the player an emote or a feat according to the NPC the player is talking to) and managed the dialogue (that we previously pointed that was just an array of dialogue lines that these components gets from each NPC parameters).

Our version of the component was heavily altered in order to comprise the part of the algorithm of the relationship model and the control of the dialogue system. As such, we will separate these explanations into those two categories:

4.4.4.A Relationship Status Calculation

The implementation of the algorithm already explained is fairly straight forward: after the player chooses the dialogue option he wants to say (with the weight attached to it), the UI blueprint will call the Interpret Function (server-side).

This function will, first, identify the structure correspondent to the player that interacted with the
merchant. After that, the evaluation function will take place. In this function, we give different weights to different criteria.

First, the personality of the merchant can contribute from 30 to 45 percent of the total perception of the option. From the remaining 70 to 85%, 35% are the memory of past interactions and 65% are the current relationship value. The final result of this will determine the new relationship status.

The personality weight is calculated like this: if both the option and the merchant sympathy traits (taken from the merchant traits table) are positive or negative, then the weight of that option is going to increase according to how high or low the sympathy level is, ranging between 30 to 45%.

After the personality weight is determined, the model will calculate the average of the past interaction scores. To do this, we access to the “Past Interaction Scores” member of the player structure (that is updated after each interaction with the interpretation of that interaction). The average calculated will fill 35% of the remaining weight of the option.

The final step to calculate the interpretation of the option and the current relationship value is to take into account the relationship status as it is previous to this interaction. This value will have a weight of 65% of the remaining weight of the option.

The function described above is as follows. Where ‘a’ is the option score, ‘b’ is the personality weight, ‘c’ is the current relationship value and ‘d’ is the average of past relationship values.

\[(ab) + (1 - b)(0.65c + 0.35d)\]

After the new relationship value is calculated, the player structure will be updated to match the alteration. This means that the “past interaction scores” will be updated to add this new value to the array and the “relationship status value” will be changed to this value as well.

### 4.4.4.B Merchant Dialogue System

After the new relationship status value is calculated (the process described above), the “current Tree Path” and the “full interaction record” elements of the player structure will be updated given the option chosen.

An event is then called on the UI blueprint correspondent to the dialogue, and that event will be calling functions of the "Merchant Dialogue System" again in order to decide which dialogue lines the merchant will say, which dialogue options the player will receive (if any) and, if there is any quest going on. These functions are now running client-side, because there is no volatile information calculated in them and because the options and dialogue lines are player-independent.

The next Tree node, the player dialogue options and, if applicable, the next tree, are determined by the relationship status after the algorithm iteration takes place and those values are stored in the player
structure. The values in this structure are then used to get the merchant dialogue and the player options from the correspondent structures. For example: If after the algorithm application, the current Tree Path is "1 0" (so the player chose a positive dialogue and then a negative one), and the current Tree is “T1 Positive” then, the function that chooses what dialogue the merchant should say, will check the current tree enumerator entry from the player structure, confirm that it is “T1 Positive”, and it will convert the string correspondent to the current tree path ("1 0") to a valid nodes’ enumerator entry and then it will play the respective merchant response from the merchant responses structure.

An iteration of a interaction of the player, where he chooses a dialogue option can be seen in the Fig.4.7

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4.7.png}
\caption{Visualization of the client-server communications}
\end{figure}

### 4.4.5 Shops

As mentioned before, to make our merchant work as intended, we had to improvise and reuse some assets already present in Conan Exiles. More specifically, to implement the shop, we found out that, in terms of functionality, it was rather similar to a crafting station. We will now briefly explain how this stations work:

- Each type of crafting station has associated to it a group of recipes (dependent of the crafting station itself and if the player has learned those recipes).
• Each is composed by the necessary ingredients to produce it, their quantities, and the outcome of the recipe.

• Other options can be added to recipes, like the need of a catalyst material or the time it takes to produce the results of the recipe.

Knowing of this, to create a shop we basically duplicated a crafting station and changed the associated blueprint to implement the following:

• The recipes are now items for sale, they can be locked and unlocked dynamically depending on triggers like the relationship status.

• Each item, instead of needing ingredients to be made, now has a price. The major difference in this is that instead of needing cumulative ingredients, now the player may choose a currency from the ones accepted to buy a certain item. For example, to trade a sword, instead of needing 30 iron and 10 wood, the player can now use 30 coins or 50 meat to get it.

To setup a shop, we need to add a new crafting station in the Crafting Station Name dataTable. In here we specify the shop name and its ID. After the creation of the shop, we need to add the items we want to sell in each shop to the Recipes’ Table and associate the correspondent recipes in theFeat Table in order to determine when the player gains or loses the ability to buy each item.

There are some problems and limitations with this: first, we can’t change the inventory of a shop easily, since it is all dependent on the players’ feats, there is no easy way to add and remove a lot of feats without a relevant amount of authoring and condition tuning. As such, we used the feat system to add and remove a few items in the shop according to the relationship value, but for other features, we ended up using more shops.

So, there is one main shop, where the player can buy items, and this shop adds and removes more items, as mentioned before, according to the relationship value. There is also an “offer” shop, this has the same UI of a main shop, but it has the purpose of allowing the player to offer items to the merchant, currently only two items can be offered and they are in direct response to a quest given by the merchant (more on that in chapter 5). Lastly, a third shop was made to be in the basic merchant. This shop doesn’t update its inventory contents or prices and it’s used as a means of comparison in the test environment (we will talk more about this in chapter 5).

Another problem comes due to the fact that the data tables are not editable in runtime, this means that we can change prices by multiplying the value taken from the recipe datatable but we can’t change the value directly in the table, so, when a function that we don’t have access to (because they are defined in the C++ side of the game) accesses the dataTable, the values are always going to be the originals. This affects us directly, for example, when the UI Icon in the shop shows the “price” of an item, the
function that determines the number showed on the icon, can’t be altered, because it is read-only to the devKit users. As such, despite the fact that the price itself can be possible to change, the number displaying that price is always the same. We implemented an workaround to this problem by printing, in a more evident font, the “true” price of an item, in case this is different from the one in the dataTable.

### 4.4.6 Multiplayer Implementation

The mod we created, was made with a multi-player component in mind. In here, we will talk about the objective behind this, the way it was implemented, and why it didn’t come through. Our idea was for multiple players to be able to interact with the merchant. The latter would remember each one of them, and present the shop according to the relationship of each one. In terms of implementation there were some basic practices we wanted to make sure were met from the start:

- All sensible information is stored, changed and consulted server side. This has to happen to prevent payers from changing their relationship status or any other kind of information. This is relatively easy if the information is stored client-side (in the players’ computer). So, to prevent such thing from happening, we make sure all the relevant information is in server-side.
- All the information is stored, not only server-side, but in the merchant instead of the player. This was done too make it easier to access the information because it is all centralized and even for testing and checking metrics it is more practical to have all that information in one place.
- There can’t be information switched between players. The merchant has to make sure to access the right player each time an interaction is started.
- The relationship information of each player should be saved even after the player disconnects from the game.

With this in mind, we designed a solution where the merchant, server-side, had an array of player relationship structures that, when a player interacted with him, he would check that array to see if there is any structure with the ID matching that player. If there is, then the model iteration happens on top of said structure (otherwise, a new structure is created, changed and added to the array).

The implementation of this array was relatively simple, the relationship structure itself already has the all the information necessary in order to check previous interactions and relationship status. With that said, it was just a matter of creating the array, and then a function that, upon interacting with the merchant, this would check if it is the first interaction of the player or not (see Fig.4.8). A thing to keep in mind when dealing with player identification is that, despite each player having a Character Unique ID value, that value itself is not the actual unique ID. If we were to compare two Characters Unique ID values with a simple equals function, the result might be negative even if the player is the same, the
Figure 4.8: Verification of past interactions between the merchant and the player

catch is that we need to use a specific equals function that receives two Unique IDs and determine if they belong to the same player or not.

As we already mentioned at the time of the development, only components that inherited from certain classes could make calls to functions server-side. Fortunately, the Merchant Dialogue Component can make those calls, but unfortunately, every component related to the shop can’t. As such, we created a component to works as a bridge between the shop component and the merchant, in order to make server side function calls from the shop (shown in Fig. 4.9). This means that for example, for the merchant

Figure 4.9: Class that allow us to call events server side
to start the selling process of an item, instead of calling the sell item function of the shop component directly, he has to call a function of the “merchant shop cc” component (that inherits from a component that can make calls server-side) and that function can make the call server-side of the sell function of the shop. The few functions that run client-side are the ones that are responsible for the UI, this makes sense since the UI is independent from player to player it makes no sense for it to exist server-side.

The multiplayer implementation was working as intended, but unfortunately, an update to the devKit made it so that the replication of the player relationship structure from the server to the client crashed the unreal editor. We came across very little help and the only solution in sight was to re-do this implementation without the structure. Since a major part of the model already created was dependent on that specific structure, the refactoring of the code, plus the uncertainty of this working even with that refactoring were enough reasons for us to drop the multiplayer implementation for now.
5

Evaluation

Contents

5.1 Environment setup .................................................. 59
5.2 Results ................................................................. 61
5.3 Results Conclusion ................................................... 67
With the model implemented and working, it was time to set up an environment and do tests in order to determine if the players prefer a merchant where the model is implemented or a merchant without the model. There were a total of 24 participants.

The protocol of this test consisted on each participant interacting with three merchants, one at a time. As we will see, one merchant had traits similar to a “nice” person, the other had traits emulating an “arrogant” person, and the last didn’t had the model implemented at all. The participants were given 15 minutes to fulfill an objective in a scenario with a certain merchant, after the fifteen minutes or the completion of the objective, the player would be asked to do exactly the same thing but with another merchant.

The order of the merchants tested was rotational, to guarantee that we had the same number of tests for each combination of merchants tested.

5.1 Environment setup

One of the main questions of this project is: Is it better for merchants in video-games to have a social component that affects the shop inventory? So, in order to have a base of comparison we had to create a “basic” merchant without the model implemented (to emulate the normal merchants that exist in current video-games) and we had to create a scenario that incentivizes the interaction with the merchants.

5.1.1 Objective and Merchants

Upon interaction with this merchant, after a simple greetings voice line, the player would be presented with the shop, this means there are no dialogue options or the possibility to offer items. The shop itself (in terms of inventory and prices) is static, meaning that there is no way of increasing or lowering the prices.

With that said, the tests consisted on asking for the player to buy a certain item. Once the item was bought, the test would finish. Players would test three times with a different merchant each time. The merchants created were:

<table>
<thead>
<tr>
<th>Table 5.1: characteristics of each type of merchant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
</tr>
<tr>
<td>Nice</td>
</tr>
<tr>
<td>Arrogant</td>
</tr>
</tbody>
</table>

As mentioned before, the objective that was given to the testers was to buy a specific item, in this case, it was a glowing stick. That item was always present in the shop of every merchant, but in the
merchants with the model implemented, the price of the item and the possible path to buy that item would change according with the relationship that the player built with the merchant.

The reasoning behind the creation of two merchants with different (and opposite) values in their traits, wasn’t to compare them, but to see if both kinds of merchants (nice and arrogant) were an improvement when compared with the basic merchant. The different merchants guarantee that the player has different experiences that can make reaching the objective an easier or harder task when compared with the basic merchant, because:

• when dealing with the "nice" merchant, the player can finish the objective faster due to this merchant easily developing a positive relationship with the player (product of having a high value of sympathy) and, when lowering the prices, the deals are very favorable for the player (because it has a low value of greediness)

• when interacting with the "arrogant" merchant, the player is presented with a harder challenge because it is difficult to increase the relationship value and easy to decrease it and, even when the player manages to reach a point where the merchant cuts the prices, these cuts are lower than the ones made by the nicer merchant.

As we mentioned before, we gave the merchant the possibility to ask for items when the player finished one dialogue tree. This had the objective of incentivising exploration and elongate the experience. The items asked could be found near the merchant location and the merchant would remind the player of this quest and point out to the locations of said items every time the player interacted with him while he was waiting for quest to be completed.

### 5.1.2 Scenario

Being a survival game and with an arguably high difficult curve, Conan Exiles wasn’t properly the easiest game to give to a tester that was playing it for the first time. The risk of the tester failed to finish the object because of a variety of reasons from dying of dehydration, falling of cliffs or just wander around and get lost. To prevent all of this, we looked for an area of the game without enemy spawners and built walls giving the player a limited test space. After that, we spawned a booth where the merchant would be, to make it easier for the player to identify the merchant from far away. Lastly, we made it so the player was invincible, because it was really easy for the tester to fall from a cliff to his death.

After securing the safety of the player, we spawned the necessary items in order to complete the merchant quests and for the player to have currency to buy the items required to finish the test.

Each test had a duration of fifteen minutes, or until the player bought the necessary item that was determined to be the objective.
5.2 Results

In this section we will see and interpret the results of the questionnaires presented to the testers. This should give us information about the success of this work, as well as pointers of what and where to improve. There were two types of questionnaires given to the testers, one type after each player interacted with each merchant (referring to said merchant), and one after all interactions (referring to the experience as a whole).

5.2.1 Merchant Questionnaire

As previously mentioned, this questionnaire was given to every tester after he completed a test with a merchant. As such, each tester filled three of these questionnaires (with the same questions). We used a 7 point likert scale, where 1 means “Strongly Disagree” and 7 “Strongly agree” for all the 19 questions. With the questions gathered we were able to create 3 groups to be evaluated: Enjoyment [43], Engagement [44] and Believability [45] [46].

The groups were decided by calculating the Cronbach’s Alpha value in order to determine the reliability of each group. Some changes had to be done to make sure that each group is reliable. This is the reason why some of the questions had to leave their correspondent group.

Despite this, upon analyses we can verify that every question had a good result that helped us confirm that the objectives proposed were met.

For the sake of simplicity, we identify the merchants, in the tests, as: Basic for the one without the model implemented, Good for the one with high sympathy and low greediness, and Bad for the one with low sympathy and high greediness.

The results themselves were, in general, very positive. The analysis consisted of calculating the means of each group results and, due to the nature of the tests, run the Wilcoxon signed-rank test, to show the levels of significance between the basic merchant and the good merchant and between the basic merchant and the bad merchant.

5.2.1.A Enjoyment

<table>
<thead>
<tr>
<th>Table 5.2</th>
<th>Table 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enjoyment</strong></td>
<td><strong>Significance Value</strong></td>
</tr>
<tr>
<td>Basic</td>
<td>2.88</td>
</tr>
<tr>
<td>Good</td>
<td>6.29</td>
</tr>
<tr>
<td>Bad</td>
<td>6.07</td>
</tr>
</tbody>
</table>
These results show us that the testers show significantly higher levels of enjoyment when interacting with either the good or bad merchant in relation to the basic merchant. We assume that this is mostly due to the higher range of interactions and reactions that exist in the non-basic merchants.

### 5.2.1.B Engagement

<table>
<thead>
<tr>
<th>Engagement</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>1.79</td>
<td>1.5</td>
<td>0.91</td>
</tr>
<tr>
<td>Good</td>
<td>5.85</td>
<td>6.00</td>
<td>0.92</td>
</tr>
<tr>
<td>Bad</td>
<td>5.83</td>
<td>6.00</td>
<td>1.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Engagement</th>
<th>Significance Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good - Basic</td>
<td>0.00002</td>
</tr>
<tr>
<td>Bad - Basic</td>
<td>0.00002</td>
</tr>
</tbody>
</table>

These results show that the levels on engagement of the players that interacted with the merchants with our model are significantly higher when compared with the basic merchant.
5.2.1.C Believability

<table>
<thead>
<tr>
<th>Believability</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>2.76</td>
<td>2.67</td>
<td>0.82</td>
</tr>
<tr>
<td>Good</td>
<td>6.06</td>
<td>6.17</td>
<td>0.58</td>
</tr>
<tr>
<td>Bad</td>
<td>5.95</td>
<td>6.00</td>
<td>0.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Believability</th>
<th>Significance Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good - Basic</td>
<td>0.00002</td>
</tr>
<tr>
<td>Bad - Basic</td>
<td>0.00002</td>
</tr>
</tbody>
</table>

![Box plot showing believability scores for Basic, Good, and Bad merchants.](image)

**Figure 5.3: Believability of each merchant according to the players**

The values taken from these tests show us that the non-basic merchants are significantly more believable than the basic one we can see in the graph 5.3 that the means of the believability values are very high in both merchants that have the model implemented, and very low in the basic merchant. This is one of the hypotheses proposed in the section 1.2.

5.2.1.D Repetitive

There was one question made in this questionnaire that wasn’t based on any other questionnaire, being that we wanted to know the answer to that particular question because we felt that was relevant given the context of the tests and the model itself.

The question made was: “Was the merchant repetitive?” and we will now show the results.

<table>
<thead>
<tr>
<th>Repetitive</th>
<th>Mean</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>6.38</td>
<td>1.41</td>
</tr>
<tr>
<td>Good</td>
<td>3.33</td>
<td>1.37</td>
</tr>
<tr>
<td>Bad</td>
<td>3.50</td>
<td>1.44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Believability</th>
<th>Significance Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good - Basic</td>
<td>0.0009</td>
</tr>
<tr>
<td>Bad - Basic</td>
<td>0.00013</td>
</tr>
</tbody>
</table>
This results show us that the model implemented did a good job in keeping the interactions fresh, despite the dialogue trees repeating in some instances, measures were taken in order to avoid the interaction from being too repetitive.
5.2.2 Final Questionnaire

This questionnaire was filled by each participant after all the interactions were done. The questions themselves were made by us and cover subjects that we determined relevant to ask given the model created and its hypotheses.

It is relevant to mention that, when we talk about "requests of the merchants" this refers to the mini-quests given when the player reach the end node of a dialogue tree.

From the data collected we can see that, despite the fact that the model we implemented was an
abbreviation of the proposed one, the results show that the players think that the features implemented in the model improve the experience.

We also asked two questions about the preferred merchant to each participant, and the results were equally positive. The point is not to compare the "good" with the "bad" merchant but to compare the merchants with the model versus the one without the model.

As stated before, despite being interesting to compare the two types of merchants who implement the model with one another, we leave that for possible future work. With that said, what we can conclude from these two questions is that a relevant majority of the participants prefer a merchant who has the model implemented when compared with one who doesn’t have one.

This leads us to believe that, with a more complete implementation, the results would be even better, because they are very positive with just a sample of the proposed features.
5.3 Results Conclusion

From the results presented, it is clear that, given the hypothesis created, this work was a success. It shows that the players prefer interacting with a merchant that has a social component within him. It is something worth investing in, because, unfortunately, there were features that had to be left out and we believe those changes could improve a lot the results.

The players seem to be open to interact with social merchants. The results showed a disparity too relevant to be ignored, although some problems were found (like the difficulty to balance the features of a merchant, so as to not ruin game flow). These can be fixed with future work.

A very surprising result was that the participants preferred the merchant with the social model even when the prices were higher when compared with the basic one. This is something very interesting because it points to the fact that the interactivity and sociability were value higher than just completing the objective. In other words, the participants didn’t mind working more towards an objective if the merchant had the model associated to him.

To sum it up, the results show that this work is a step in the right direction when it comes to player interaction, immersion and experience in video-games, and it satisfies the goals proposed in the chapter 1.2, despite having room for improvement.
In this work, we proposed to tackle some limitations in current video-games. The fact that, in a big majority of games, the NPCs that have the role of merchants, do not have any kind of engagement to the player, they feel like a tool to implement a trading mechanic, and not a character with a personality and social characteristics.

Other problem recognized was that associated with NPCs with any depth, there is a lot of authoring behind it.

With this in mind, we developed a model that aims to solve or minimize of those problems, specifically in NPCs that serve as merchants.

The model had to fulfill some requirements. It had to be generic enough to be possible to implement in a variety of games, it had to reduce the authoring effort (when compared with the creation of a social NPC with the same characteristics has the ones we want to make) and it had to improve the believability of the character and overall player experience.

The model was implemented using the Conan Exiles DevKit, that is a modified version of the Unreal Engine. We choose this game because it is a recent commercial and successful game with the necessary tools to be worked upon (modded). The mod created gives the possibility to the developer of creating a merchant with two different characteristics: sympathy and greediness. A value from 0 to 10 can be attributed to those characteristics in order to generate a custom merchant with a unique behavior.

This helps reduce the authoring, because changing the behavior of the merchant requires just the alteration of variables instead of manually changing every new cation wanted.

According to the results of the tests done, 93% of the players showed preference of the merchants with the model implemented when compared with the merchant without the model. On the same note, when questioned about the enjoyment and believability, the preference for the merchants with the model was very significant. Another accomplishment was the fact that the results, the model implemented did not compromise the flow of the interaction between the player and the merchant.

The outcome of this work show very promising results, it is worth to dedicate more resources in this area, because a lot of times it's pushed aside when developing a video-game, but it is proven that with some investment it has the potential to improve substantially the experience of the player when playing the game.

6.1 Future Work

There are a lot of improvements to be done to this work. One think our model is prepared to do is to be expanded, this means that a lot of experiments can be done with new traits (being normal traits or merchant traits) and other features.
The implementation of the merchant and player preferences is something worth working on, because it is a system that complements well the model implemented.

Since the model capable of being integrated in several types of games, it would be interesting to increase the nature of the outcomes. The result of an interaction can be more than just changes in prices or a different dialogue reply, it can be, for example, the trigger for a quest or something more outside of the box.

Features such as traits or components added to the model can give extra layers to the interaction and complexity of the merchant. Of course that, despite the model being done such as the authoring effort is minimal, it always increases with the number of variables added and the balancing of those variables can become a problem.

As stated in the section 4.4.6, the multiplayer implementation, despite being playable at a certain point of the development, had to be dropped. It could be interesting to apply these merchants in a multiplayer environment and see how the different players interact with them.

Another point that could have been more worked on was the decision making module. Ideally, this module would have a relatively simple interface where the developer could state what are the possible outcomes of a merchant interaction and what the conditions for them to happen. Unfortunately, due to the problems mentioned before, such a interface was not possible to be developed, it is certainly something that could improve the portability of the model and worth working on.

To sum it up, there are a lot of angles from which to improve this work, being the majority of them in the form of additions and expansions to the features of the model.
Bibliography


A.1 DataTables

- **Merchant Traits:** Contains the normal and merchant specific traits’ values. In our base, we have the Sympathy, Greediness and a boolean stating whether the merchant has the model applied or not (we do this for testing purposes).

- **Merchant Dialogues:** Has the reaction dialogues (sound files) for each merchant.

- **Crafting Station Name:** Has the IDs and names of the modified crafting stations we used to make the shop and offer environment.

- **Recipes Table:** This table dictates which recipes (Items) will be in each shop. Each entry has to be filled with the materials necessary (price) and the result of the purchase (the item itself).

- **Feat Table:** In our implementation, we use feats as a way to specify “bundles” of items that can be unlocked, which means that we group certain items as a reward of a feat, and when that feat is unlocked, those items become available at the shop.
• **Spawn Data Table:** This is the table where we add the merchants we want to spawn. In here we specify the NPC Class (in our case it’s Dialogue Merchant) and we specify the parameters (that allow us to link the NPC to the correspondent dialogue tree).

• **Player Options Tree:** This table contains all the text dialogue that the player can say to each merchant, organized by trees and nodes.

• **Merchant Responses Tree:** Similar to the player options tree table, but this has the sound files correspondent to each merchants response from the dialogue tree.

### A.2 Information flow of an interaction

In this section we will try to present the information flow of a basic interaction. The action that the player chose in this example was a positive dialogue said to a merchant with a high value of sympathy and a medium value of relationship status between both:

1. The player approaches the merchant with a positive dialogue option, with the value of 8.25 (from 1 to 10, where below 5 is considered a negative action and above 5 a positive one). This action happens client side, in the UI blueprint and a “bridge function” will be called in order to run the Interpret Option Function in server-side (with the value of the option chosen). This can be seen in Fig.A.1.

![Figure A.1: Positive Dialogue Chosen](image)

2. The Interpret Option Function will be called server-side and this function will begin by identifying the player interacting and return the relationship struct (stored in the merchant array of structures) as shown in Fig.A.2.

3. Now, the evaluation function will take place in order to get the new relationship Structure. This process will, first, determine the weight that the personality of the merchant has (Fig.A.3). In this case, let’s say that the merchant has a sympathy value of 10 (max), this means that, since the action chosen is positive and it’s value is above the current relationship status, the merchant will weight that option as 45% (max) of the new relationship value (Fig.A.4).
4. With the merchant personality weight found, in this case 45%, the remaining 55% of the result will be split as such: 35% will be filled by the means of the last 30 relationship values (for the sake of simplicity, we can say that it is 5 in this example). The other 65% are correspondent to the last
relationship value (before the current interaction), in this case we can say it’s 7.

5. Having the new relationship status value:

\[(0.45 \times 8.25) + (0.19 \times 5) + (0.36 \times 7) = 7.18\]

the *Interpret Option Function* will now update the structure of the player with this new value and will calculate the next tree node and, if applicable, tree (the calls to these functions can be seen in Fig.A.5).

![Figure A.5: Relationship Structure update](image)

6. Next, a client event will be called in order to deal with updates on the UI and/or shop inventory. This event receives the new relationship value, and information regarding the position of the player in the dialogue tree (this is to check if there is any “quest” to be triggered, because in our case, we use the end nodes of each tree to trigger quests). This set of conditions and verifications can be seen in Fig.A.6).

![Figure A.6: Call to client side functions after determination of the new relationship value](image)
7. The function to determine which line of dialogue the merchant will say takes place on the client-side. This function checks the tree and node that the player is currently in, and plays the sound correspondent to it (this runs on the merchant dialogue component). The Fig. A.7 shows the function getting the current tree and node while Fig. A.8 shows the general view of all the possible dialogue lines to be picked.

Figure A.7: Determination of the current tree and node
8. Afterwards, there will be a verification to see if there is a quest triggered. If so, the dialogue buttons disappear and, if applicable, items will be added to the offer UI.

9. The player options will be determined next (if there is not a quest going on). The principle is similar to the choice of the merchant dialogue.

This was a basic example of a interaction between the player and the merchant. The relationship status between both changed after that interaction, and that may translates in changes in the shop itself. Those changes are only calculated when the player opens the shop, like this:

1. The player presses the open shop button in the UI, this will call the update shop function with the relationship value between the interacting player and the merchant(Fig.A.9).
The update shop function will do two major things: first, it will check the relationship value in order to determine the price variance and the items that are locked or unlocked in the shop. In this example, if the relationship status is above 8, then the price variation will be 0.6 (40% discount at best) and the item resulting from the recipe 101 will be unlocked as can be seen in Fig. A.10.

The second alteration this function will do is change the price variation according to the merchant treats (in this case, let’s say the greediness of the merchant is 1, which means he is not greedy at all) the discount will be positively influenced, so, instead of 40%, the discount may be 35%.
3. After the shop is opened, the dialogue of the merchant (reaction dialogue) will be chosen and the UI will be updated (Fig. A.11).

**Figure A.11**: Functions that complete the opening shop process