TÉCNICO LISBOA

From Caveman to Gentleman

Implementing an artificial intelligence social architecture in Conan Exiles

Luís Miguel Correia Morais

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. João Miguel De Sousa de Assis Dias
Prof. Pedro Alexandre Simões dos Santos

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. João Miguel De Sousa de Assis Dias
Member of the Committee: Prof. Rui Filipe Fernandes Prada

May 2019

Acknowledgments

I have never been good with words for this kind of things. I even thought of checking what other people usually write in this section of the document, but that would probably be cheating, right?

I want to thank my supervisors for supporting and helping me through this work. *Thank you for helping me grow.*

I have few but I am very grateful for them, my friends. I have to name Ricardo for accompanying me through this adventure that we call college. I know what you must be thinking: “oh, you haven’t named me”.. don’t worry, I assure you that I treasure your friendship. *Thank you for the company.*

My family has always been there for me. This is not something that say lightly, I know that I can always count on their backing, regardless any circumstances. This applies to some who are not biologically from my family. That is a mere, unimportant detail. You know who you are. *Thank you for loving me.*

Last but not least, I want to thank my love, Daniela. You have truly pushed me to the finish line. Please keep doing it for the rest of my life! I am very grateful and lucky to be with you. I hope I give you reasons to feel the same. *Thank you for being part of me.*

Abstract

Despite living in a golden age for video games where there is constant improvements being made on immersion, either through rich storytelling, high graphical fidelity or even the video game interface itself (such is the case of Virtual Reality (VR) video games) one area that is generally lacking on innovation is the Non-player Character (NPC) social interaction.

Our objective for this work is to improve the the NPCs featured in the AAA video game *Conan Exiles* [1], endowing them with behaviour that makes them more interesting from the social point of view. For this, we based our solution on the Comme il Faut (CiF) architecture [2], specifically focusing on the Comme il Faut - Creative Kit (CiF-CK) [3] architecture and extending its features, adding emotions and a emotion appraisal process based on the OCC model of emotion and also a belief system that models the social network values that the NPCs expect for the other NPCs' relationships.

Our model, called the Comme il Faut - Exiles (CiF-Ex), was validated by user tests. The results were noteworthy: the users enjoyed more the game and tended to spend more time near CiF-Ex enabled NPCs. The majority of the users preferred CiF-Ex NPCs, even without voice acting versus the vanilla NPCs with voice acting. CiF-Ex NPCs were perceived as more believable and less predictable.

This document describes in detail the process of research, idealization, implementation and test of the *CiF-Ex model*.

Keywords

Social Model, Non-player Character, Social Interaction, Artificial Intelligence, Video game.

Resumo

Apesar de vivermos numa era dourada para os videojogos em que se aposta cada vez mais em imersão, seja através de narrativas ricas, da elevada fidelidade gráfica ou até mesmo da interface do jogo (caso dos jogos de realidade virtual), a área de interacção social com NPCs (non-player characters - personagens não jogáveis) não tem visto grande inovação.

Desta maneira o nosso trabalho procura melhorar os NPCs no videojogo AAA *Conan Exiles* [1], dotando-os com comportamentos que os tornem socialmente mais interessantes. Para isto baseámos a nossa solução na familia de arquitecturas *Comme il Faut (CiF)* [2], especialmente focando-nos na arquitectura *Comme il Faut - Creative Kit (CiF-CK)* [3] e alargando as tuas características, adicionando emoções e um processo de apreciação de emoções baseado no modelo de emoções OCC e também um sistema de convicções que modela a espectativa de cada NPC relativamente aos valores das redes sociais dos outros NPCs. O nosso modelo, chamado de *Comme il Faut - Exiles (CiF-Ex)*, foi validado através de testes de utilizador. Os resultados obtidos foram muito favoráveis: os utilizadores desfrutaram mais do jogo e tenderam a passar mais tempo perto dos NPCs com o CiF-Ex activo. A maior parte dos utilizadores preferiram os NPCs CiF-Ex, mesmo não tendo dobragem de voz como os NPCs originais. Os NPCs CiF-Ex foram captados como mais credíveis e menos previsíveis.

Este documento descreve em pormenor o processo de pesquisa, idealização, implementação e teste do modelo CiF-Ex.

Palavras Chave

Modelo Social, Personagem Não Jogável, Interacção Social, Inteligência Artificial, Videojogo.

Contents

1 Introduction **1**

1.1 Objectives . . . 3

1.2 Outline of the document . . . 4

2 Related Work **5**

2.1 Conan Exiles . . . 6

2.1.1 Features . . . 6

2.1.2 Conan Exiles DevKit . . . 7

2.2 Social Models . . . 9

2.2.1 *Comme il Faut (CiF)* . . . 10

2.2.1.A CiF Architecture . . . 10

A – **Social state:** . . . 10

B – **Character:** . . . 11

C – **Predicates:** . . . 11

D – **Rules:** . . . 11

E – **Influence Rule Sets (IRS):** . . . 12

F – **Microtheories:** . . . 12

G – **Effects:** . . . 12

H – **Social Exchanges:** . . . 12

I – **Trigger Rules:** . . . 12

2.2.1.B CiF Algorithm . . . 13

2.2.2 *Ensemble* . . . 14

2.2.3 *Comme il Faut - Creation Kit (CiF-CK)* . . . 16

A – **Social Exchanges** . . . 17

B – **Characters** . . . 17

C – **Social State** . . . 17

D – **Trigger Rules** . . . 18

E – **CiF-CK Implementation** . . . 18

2.2.4 *Fearnot Affective Mind Architecture (FAtiMA)* . . . 19
2.2.5 *GAMYGDALA* . . . 20
2.2.6 *PsychSim* . . . 20
2.2.7 *Synthetic Group Dynamics (SGD)* . . . 21
2.3 State of the Art . . . 22
2.3.1 A Merchant Model . . . 22
2.3.2 A Procedural quest generator for Conan Exiles . . . 23
2.3.3 Façade . . . 24
2.3.4 The Sims . . . 24
2.4 Discussion . . . 26
3 Comme il Faut - Exiles **27**
3.1 Solution . . . 28
3.1.1 Social Exchanges . . . 30
3.1.2 Characters . . . 31
3.1.2.A NPC Cycle . . . 34
3.1.3 Social State . . . 34
3.1.4 Microtheories . . . 35
3.1.5 Trigger Rules . . . 36
3.2 CiF Cycle . . . 36
3.2.1 Desire Formation . . . 36
3.2.2 Intent Formation . . . 37
3.2.3 Perform Social Exchange . . . 37
3.2.4 Performance Realization . . . 38
3.2.5 Social Fallout . . . 38
4 Implementation **39**
4.1 Social Exchanges . . . 40
4.1.1 CiF Dialogue Widget . . . 42
4.2 CiF NPCs . . . 45
4.2.1 Initialization . . . 46
4.2.2 General Behaviour . . . 47
4.2.3 Social Attributes Structure . . . 48
4.3 CiF GameState . . . 50
4.3.1 CiF Cycle . . . 52
4.4 Challenges and Difficulties . . . 53

5 Evaluation **57**
5.1 Scenario . . . 58
5.1.1 Control Condition - Conan Exiles Dialogue NPCs . . . 58
5.1.2 Experimental Condition - CiF-Ex NPCs . . . 59
5.2 Procedure . . . 61
5.3 Metrics . . . 62
5.3.1 User Experience . . . 62
5.3.2 NPC Believability . . . 62
5.4 Results . . . 62
5.4.1 Questionnaire answers . . . 63
5.4.1.A Profile Questionnaire . . . 63
5.4.1.B Metrics Questionnaire . . . 64
5.4.2 Data Analysis . . . 64
5.4.2.A User Experience . . . 65
5.4.2.B NPC Believability . . . 66
5.4.3 Discussion . . . 67
6 Conclusion **69**
6.1 Future Work . . . 71
A Code **77**
B Tables and Figures **79**

List of Figures

2.1 Conan Exiles screenshot . . . 6
2.2 Example of a blueprint . . . 8
2.3 Basic structures of CiF [4] . . . 10
2.4 Interaction between core CiF components [5] . . . 13
2.5 CiF algorithm [2] . . . 14
2.6 Diagram of a social practice . . . 15
2.7 The internal structure of CiF-CK [3] . . . 19
2.8 FAtiMA's simplified architecture [6] . . . 20
2.9 PsychSim's interface [7] . . . 21
2.10 The SGD model in the mind of each member of the group [8] . . . 22
2.11 The merchant's architecture [9] . . . 23
2.12 The quest system architecture [10] . . . 23
2.14 The Sims 4 screenshot . . . 25

3.1 The CiF-Ex Components . . . 29
3.2 CiF-Ex NPC Cycle . . . 34
3.3 The CiF Cycle . . . 36
3.4 The CiF Cycle and CiF-Ex Components . . . 38

4.1 Player to NPC interaction widgets . . . 44
4.2 CiF Behaviour Tree . . . 48
4.3 Social Exchange interaction messages . . . 49
4.4 CiF Dialogue NPC Info Widget . . . 50
4.5 CiF Cycle Event . . . 52
4.6 The UE4 interface for editing datatables . . . 54
4.7 Exported datatable to CSV format . . . 54
4.8 Excel file for the Dialogue Responses . . . 55

5.1 User study spawn . . . 59
5.2 Study's Trait values . . . 60
5.3 Study's Friendship Social Network values . . . 60
5.4 Study's Admiration and Attraction Social Network values . . . 60
5.5 Profile questionnaire pie charts. . . . 63
5.6 Cronbach's Alpha results for all the metrics and conditions . . . 64
5.7 Mean bar chart for the *Enjoyment* . . . 65
5.8 Mean bar chart for the *Flow* . . . 66
5.9 Mean bar chart for the *Bonfire Time* . . . 66
5.10 Pie chart for the *Condition Preference* . . . 67
5.11 Mean bar chart for the *Believability* . . . 67

B.1 Stacked chart of answers for Flow . . . 86
B.2 Stacked chart of answers for Believability . . . 87
B.3 Stacked chart of answers for Enjoyment . . . 88
B.4 Example of Weights for the Influence Rules . . . 90

List of Tables

2.1 Relation between CiF and CiF-CK Characters Social Exchanges [3] 17

2.2 Relation between CiF and CiF-CK Characters [3] 18

2.3 Relation between CiF and CiF-CK Social State [3] 18

3.1 Examples of the Social Emotion Status 32

3.2 Examples of the Emotion Status 32

3.3 Examples of the Physical Status 33

3.4 Social State in CiF-Ex 34

B.1 Social Exchanges in CiF-Ex 80

B.2 Short description of the Social Exchanges 81

B.3 Characters in CiF-Ex 82

B.4 Short description of the Traits 82

B.5 *CIF_SocialExchange_Component* 83

B.6 *Struct_CIF_SE_Attributes* 83

B.7 *W_CIF_NPCDialogue* 84

B.8 *BP_CIF_NPC* 84

B.9 *Struct_CIF_Social_Attributes* 85

B.10 *BP_CIF_GameState* 85

List of Algorithms

A.1 CiF-Ex Cycle pseudocode 78

Acronyms

CiF	Comme il Faut
CiF-CK	Comme il Faut - Creative Kit
CiF-Ex	Comme il Faut - Exiles
MT	Microtheory
NPC	Non-player Character
RPG	Role-playing Game
SE	Social Exchange
SN	Social Network
VR	Virtual Reality

1

Introduction

Contents

1.1 Objectives 3
1.2 Outline of the document 4

Artificial intelligence is slowly making its way into our lives. Recent advancements in AI have allowed technologies like Apple's Siri [1], Microsoft's Cortana [2] or Google's Assistant [3] (all mobile assistants with a degree of personality and sociability) to get into the pocket of the mainstream person. This suggests that the general public is open to interact with and integrate social artificial intelligence in their daily lives.

In parallel, the video games industry has seen a growing concern in representing the world as we know it in order to give the users a more engaging experience. This attentiveness to realism manifests in different forms - we can observe the constant improvement over the years in graphic fidelity; analyze the enthusiasm that Virtual Reality caused with the appearance of the Oculus Rift [4] and consequent race to release VR headsets between companies like HTC with the HTC Vive [5] and Sony with Playstation VR [6]; and consider the trend in current video games to grant rich stories with distinct characters. In short, greater realism and interactivity are two targets to pursuit.

In this context, AI can be seen as an augmentation to the gameplay experience. Despite the abundant number of AI controlled characters in AAA games nowadays, Human-AI social interaction in video games is usually very simple, devoid of characterization and consequences, which creates a contrast between the individuality that those characters display in cutscenes and the actual gameplay.

An example that illustrates this point is The Witcher 3 [11], a video game that is highly regarded for its storytelling and compelling world building. The story NPCs have rich dialogues with the protagonist, Geralt of Rivia and there is even branching dialogue. However the immersion caused by this impressive behaviour is broken when the player realizes that it only happens for highly scripted cutscene like interactions during quests. Outside of those, the same important NPCs will have the same scripted branching dialogue over and over.

Most NPCs will respond with the same sentences over and over or if they are more relevant to the world, like traders or named NPCs, they will have scripted dialogue options that will repeat in future contacts. The reappearing dialogue lines can be fixed with additional focus on creating new lines, but those would still be far from meaningful interactions that impact the NPCs on a social level.

Social interactions that cause real consequences in the game world are hard to model. As the player is given more freedom of social choice and more detail is added to the characters, the space of possible social consequence grows rapidly. If the developer chooses to represent the player's choices with a traditional branching tree, they would have to account for all the possible combinations of player actions, meaning that the amount of content that needs to be created would grow accordingly. Consequently the studios would have to allocate more resources. In short, better social behaviour increases significantly the authoring burden. This issue is usually referred as "the authoring problem". To combat this tendency,

[1] https://www.apple.com/ios/siri/
[2] https://www.microsoft.com/en-us/windows/cortana
[3] https://assistant.google.com/
[4] https://www.oculus.com/
[5] https://www.vive.com/eu/
[6] https://www.playstation.com/en-us/explore/playstation-vr/

new, non-conventional models that dynamically adjust the social panorama have to be applied.

To accomplish our goals we also need a system that shows potential to be enhanced with new social abilities and that provides the necessary tools to perform those modifications. *Conan Exiles* [1] is an open-world survival video game published and developed by Funcom. This video game features a giant sandbox world whose characters have simple social interaction and also provides modding tools to modify the base game, the *Conan Exiles DevKit*. As the Conan Exiles' social component is basic, this game could be used to benchmark a social model.

1.1 Objectives

Our objectives for this thesis are to improve the **User Experience** and the **NPC Believability** for the AAA video game *Conan Exiles*.

- **User Experience:** The *User Experience* is a tool to quantify how engaged the players are in the video game. Various elements contribute to the *User Experience*, such as *flow*, the interface and controls. Considering that video games are a product, the *User Experience* is very important to maximize as it is an indication of the overall quality of the video game, dictating how the consumers will judge the product, or in other words, its success. **We expect the players to *enjoy* interacting with the NPCs and explore their capabilities, in turn improving the *User Experience***;

- **NPC Believability**: *Believability* measures the user's perception of the NPCs compared to the expectation that they have for the said NPCs. Characters are part of a world and need to live within it displaying a coherent presence, not only without disrupting the player's immersion but also helping to improve it. **As we are adding new capabilities and dynamism to the social behaviour of the characters, we aim to achieve NPCs that are more *believable***.

A family of architectures for social AI, CiF [2], has had success with its implementation in Prom Week [7] and with a more recent extension of the architecture, CiF-CK [3], which was integrated on *The Elder Scrolls V: Skyrim* [12] as a mod (fan-made alterations to video games that add or modify mechanics). One of the characteristics that distinguishes this system from others is that it has been associated and implemented in video games, while others are normally used in demos, designed to showcase the architecture itself.

With this in mind, our hypothesis is that **integrating a social model on top of Conan Exiles** will improve the *User Experience* and the *NPC Believability*.

[7] https://games.soe.ucsc.edu/project/prom-week

1.2 Outline of the document

The organization of this document is as follows: **Chapter 2**, **Related Work**, has an overview of some the research that has been done on social models and some implementations. The current social capabilities of the *Conan Exiles NPCs* and the available tools to modify the game will also be scrutinized. This analysis will allow us to form a conjecture of a viable solution for our goals. In **Chapter 3**, **Solution**, we describe the social architecture that we will implement. **Chapter 4**, **Implementation**, presents a technical view of the implemented system. In **Chapter 5**, **Evaluation**, we define the used methodology to validate the goals that we have set to achieve in the current Chapter as well as the setting used to test the architecture. The last chapter, **Chapter 6**, **Conclusion**, has a summary of what we have achieved and some considerations on future work.

2

Related Work

Contents

2.1 Conan Exiles 6
2.2 Social Models 9
2.3 State of the Art 22
2.4 Discussion 26

In the first section of the Chapter 2 we give a quick overview of Conan Exiles. In the second part we describe a number of social models that can be used as a solution to the problem. In the third part we talk about state of the art projects and video games that feature interesting social NPC behaviour.

2.1 Conan Exiles

Conan Exiles is an open-world survival video game published and developed by Funcom. Conan Exiles was distributed as an early access title for the PC in January 2017 and for the Xbox One in August 2017 and released on May 8th 2018 for both platforms plus the Playstation 4. Funcom has continuously released updates that build on and improve the game's mechanics. In this section we will give more details about this video game and the DevKit, a modding tool provided by Funcom.

2.1.1 Features

The video game is set on a barbaric wasteland inspired by the world of Conan the Barbarian [13], created by Robert E. Howard. The player assumes the role of an exile in this hostile land and it is up to them to endure the many dangers of this environment, be it attacks from hostile NPCs and other players or dealing with the physical necessities of their character (hunger, thirst, sleep).

Figure 2.1: Conan Exiles screenshot [1]

The video game features mechanics to help the survivability of the players: growing crops, hunt animals or trap fish for food and gathering resources. Resources can be used to complete recipes, gained by leveling up or found in the world. Recipes are a part of the crafting system that allows players to create tools that make tasks more efficient, equipment that makes the character stronger and building parts used to create entire player-made settlements.

By default most human NPCs are hostile and attack the player on sight. It is possible to find non-aggressive named NPCs, whose purpose is to teach the player emotes (animations that the player character can perform). The player can subdue hostile characters and persuade them to become friendly by forcing them to work, breaking their will. After this procedure, the NPCs are designated by "thralls".

At first, thralls behave like an object that can be placeable on the terrain. When the player puts thralls in the world, they are assigned roles such as guard duty or blacksmithing and populate settlements.

Dialogue NPCs are named characters (like Conan himself) that have their own backstory. Most of them are priests that talk about their devotion or motivations with the player. When interacting with the Dialogue NPCs the player can:

- Choose the "Talk" option, the NPC then selects a sentence from their dialogue set and displays it.
- Choose to learn an Emote (only some NPCs have this option).
- Choose to learn a Religion (only some NPCs have this option).
- Choose to learn a crafting Recipe (only some NPCs have this option).

Despite being the one to start the conversation, the initiative is taken away from the player as they don't have the chance to react to the dialogue lines that the NPC says. The interaction is more of a monologue than a dialogue. These interactions don't have social consequences. As it stands, Conan Exiles presents itself as a testbed with potential for social artificial intelligence testing.

2.1.2 Conan Exiles DevKit

Conan Exiles was created using the Unreal Engine 4 (UE4) [2], one of the cutting-edge engines used for developing numerous video games [3]. Funcom released a modified version of UE4, the *DevKit*, downloadable in the Epic Games Launcher, granting the modding community access an environment to create and publish mods. A group of modders have created a Discord server dedicated to discussing, making and helping with Conan Exiles mods.

[1] https://zompist.wordpress.com/2017/02/06/conan-exiles/
[2] https://www.unrealengine.com/en-US/what-is-unreal-engine-4
[3] https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games#Unreal_Engine_4

The DevKit has however one limitation, despite providing most of the tools that the developers use to make the game, it does not permit modders to interact with or even consult the game's code. This means that some classes are inaccessible, can't be changed directly and their internal functionality will have to be assumed.

The assets that can be used are available through one of the features of the UE4, the system called *Blueprints Visual Scripting* [4], *blueprints* for short, designed to define object-oriented classes and objects using a graphic node-based interface.

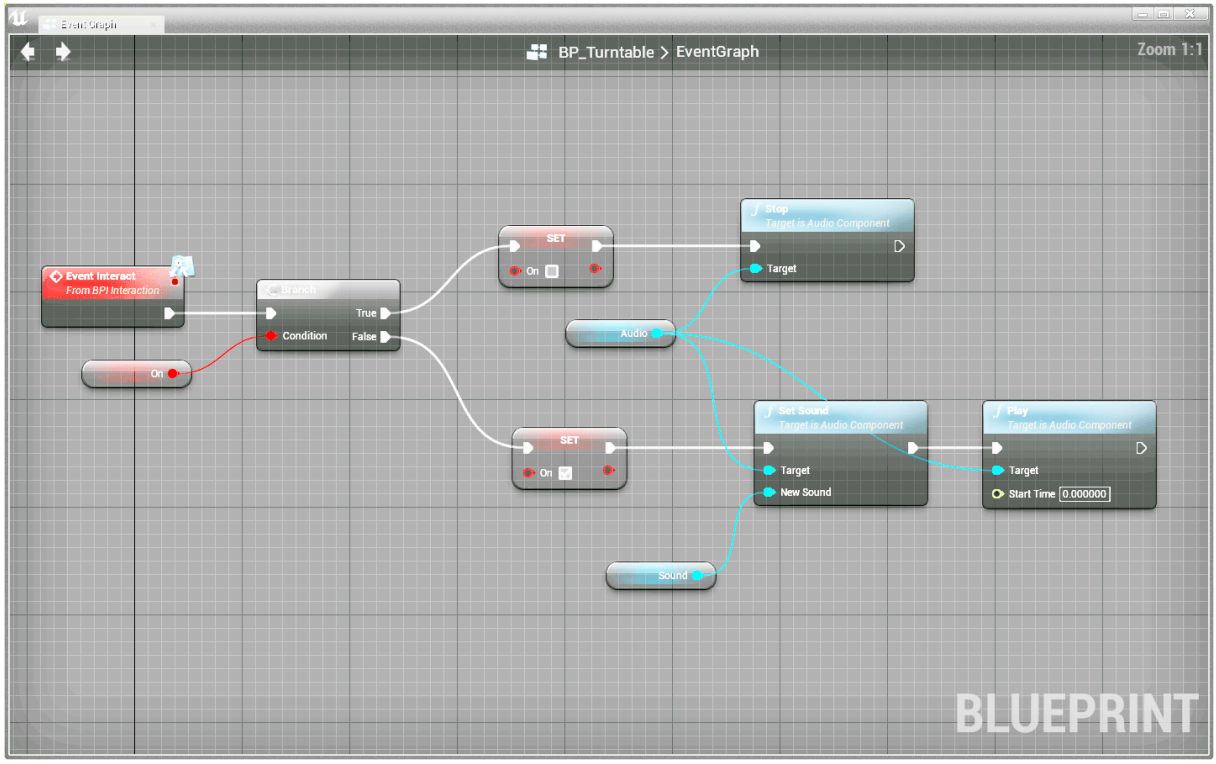


Figure 2.2: Example of a blueprint [5]

Blueprints are a visual alternative to coding and can be used to define game events, new functions, classes and instancing them. By combining different nodes, blueprints are a powerful tool to add functionality to the game. Nodes are the basic unit of the blueprints system and can be of various natures. Nodes are connected by execution wires that determine the graph's flow.

Event nodes are triggered by the game and handle the expected behaviour, so they must be in the beginning of any graph flow. Events are methods that can call other events or *functions*, *timeline* nodes (used to build simple time-based non-cinematic animations, like opening doors), *latent* nodes (like the *delay* node that stalls the flow of the event for a determinate duration of real time) and modify global

[4] https://docs.unrealengine.com/latest/INT/Engine/Blueprints/index.html
[5] https://layoutvr.github.io/learn/core_2.html

variables. Each event is executed on its own separate thread, so many run in parallel, independent of each other and the event's process ends on its own time, never hindering the execution of other tasks. This also means that we can't depend on the event's result and that events don't have a return value.

Function nodes, triggered inside blueprints, are collections of nodes that encapsulate a task. Functions can manipulate local variables and return at least one value. Functions can't use latent nodes.

Function nodes, are similar to functions, however, they don't have local variables and can't be called outside the blueprints that reference them. Macros can have more than one input or output execution wires, use latent nodes and are only compiled if used in run-time.

Nodes native to UE4 encapsulate basic functionality like controlling the flow of the graph with *while loops*, *conditional branches*, *timeouts*, etc; getting and setting values; handling type conversions (casts). Nodes can also be user created either by joining blueprint nodes on functions/events or by code (the Conan Exiles DevKit does not allow the user to create blueprints via code).

Structures are collections of data that can be of different types. The data can be accessed and modified. Structures are also used as base to the creation of *Datatables*.

DataTables are a data structure in UE4 that allows relevant information to be stored in a matrix and easily accessed. It can't however be updated while the game is running. Datatables are created using structs as a template: the struct defines the type of data that the datatable will store and the datatable instantiates pre-defined values.

Actors are objects in UE4 that can be placed in the world, be it a sword or, more relevant to this work, an NPC. They support geometric transformations like translate, rotate and scale as well as spawning and destroying operations, called using blueprints or C++ code.

Components are instanced sub-objects within actors that define the actor's behaviour or functionality. Components can be swapped in order to change the actor's behaviour while the game is running. As components are instanced, components of the same type are independent from each other and any action performed on one of the instances is only reflected on that one.

2.2 Social Models

A social model improves the NPCs' social behaviour by simulating human characteristics like traits, emotions and desires that are impacted based on the context of their social interactions. NPCs are vital to create a living world - through the simulation of human behaviour they make the world feel even more dynamic and believable.

2.2.1 *Comme il Faut (CiF)*

The aim of the *Comme il Faut* [2], social AI system is to provide a model in which the author is able to create representations of social norms and interactions that are reusable and recombinable depending on the context of the communication, while still keeping the burden of authoring social AI behaviour low. The result is an interactive social environment whose characters have distinct personalities, goals and relationships and are able to socially interact between them.

CiF lowers the authoring burden needed to create a rich and interactive social environment to tell a story by allowing the creator to specify the social context of the world in the form of rules. The nature of the rule based representation allows new domain knowledge to be added and used right away. Rather than a more simple action-reaction approach or behaviour tree, the resulting interactions are spontaneous but coherent with the foundation of the encoded social world and the involved characters.

2.2.1.A CiF Architecture

Below we will describe the structures that constitute this model as well as the algorithm that governs it.

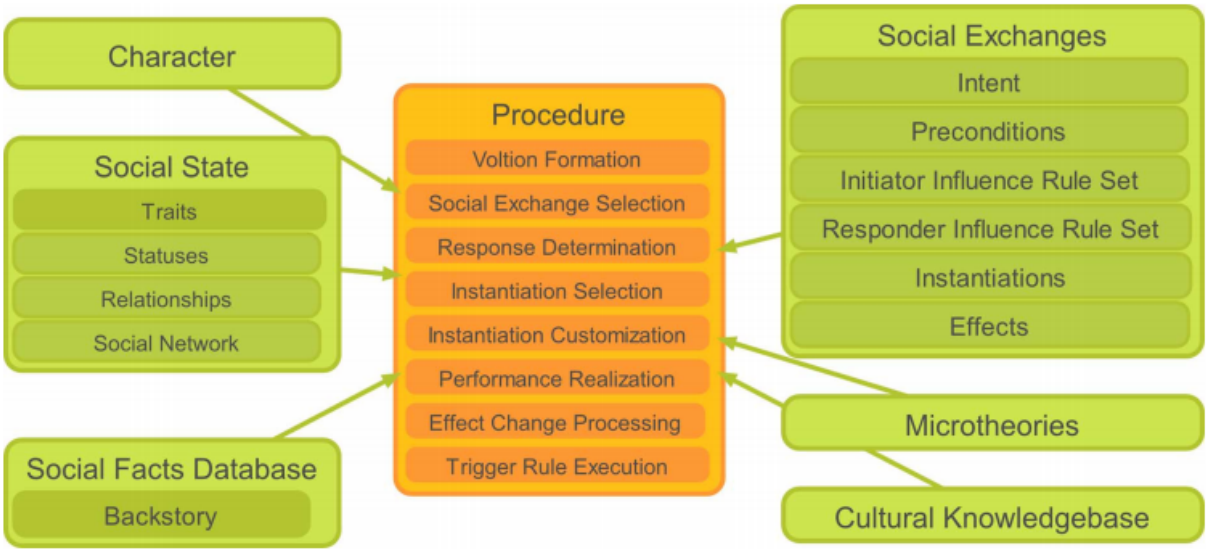


Figure 2.3: Basic structures of CiF [4]

A – Social state: a structure that represents a snapshot of the social panorama at any given time. It is composed by:

- **Cultural Knowledgebase (CKB):** a database that models the world objects. Each object is assigned a descriptive label, taken as an absolute truth in the world. For example, a *crocodile* is *dangerous*. Objects have a second label that represents the perception a character has of that

object. If one character is very powerful, they can have a contradictory perception of a crocodile and consider it *harmless* instead of *dangerous*.

- **Social Facts Database (SFDB):** a database that stores information of every single social interaction that has happened before in the world. The data stored in this database contributes to add both detail and dynamic to the world, since it can be queried and have an impact in the future. If a character spreads lies, using the SFDB they can be labelled as a liar, causing others to distrust them from that moment onwards. This way, the SFDB can be used to create the backstory of the world.
- **Relationships:** a reciprocal state between two characters (friendship, enmity, etc). This state does not change often.
- **Social Networks:** a bidirectional graph that connects every character modelling how each character perceives the others. There is one Social Network (SN) for each attitude (respect, fear, etc). These are more volatile than relationships and contribute to their change.
- **Character Status:** temporary non-reciprocal relationships that dissipate on their own (having a crush, being depressed, etc).
- **Character Traits:** static attributes that affect the results of social interactions (confident, weakling, etc).

B – Character: a structure encapsulates the social characteristics of a character. It consists of:

- **Name:** the name of the character.
- **Gender:** the gender of the character.
- **Traits:** the traits that define the character.
- **Status:** the status that afflict the character.
- **Prospective Memory:** a set of desires that the character wants to affect the world.

C – Predicates: primitives that can be evaluated to either true or false for a world in a given social state. Predicates are formed by a set of character variables, a predicate type that consists of the social environment attributes modeled by CiF, such as traits, relationships, status, social network values.

D – Rules: used by CiF to reason over the world state. Formed by predicates.

E – Influence Rule Sets (IRS): attribute a value, weight, that expresses the relevance of a specific rule in determining intents, the social change after a social exchange.

F – Microtheories: formed by a definition and a pair of IRSs. The definition is a set of one or more predicates that when true allows the Microtheory (MT) to be considered. The first IRS is relative to the initiator and the second to the receiver.

G – Effects: composed by a pair of rules, the effect conditions (must be true for the effect to apply) and the social changes (that delineate how the state of the world is changed if this effect is activated) and a label that shows if the effect was accepted or rejected.

H – Social Exchanges: the space of possible social interactions between characters. It is formed by an *intent*, a set of *preconditions*, IRSs for the initiator, *i*, and for the receiver, *r*, a set of *effects* and a set of *instantiations*.

The intent is a predicate that exhibits the change that *i* wants to make on the world. The set of preconditions is a set of microtheories that must be evaluated as true for the Social Exchange (SE) to be considered.

The IRS for the *i* and for the *r* have different contexts, the IRS weights for *i* represent how much *i* wants to perform a social exchange (*volition*). After all rules are considered, the ones with the highest score are the ones that *i* wants to perform the most. After the same process is done for *r*, they only have to either accept or reject *i*'s proposed exchange.

The set of effects should at least have two effects, one in case the case that the exchange is accepted, other if it's rejected. Instatiations are scripted dialogue acts with corresponding animations. The set of instantiations has the instantiations that the characters will play accordingly, matching the effect.

I – Trigger Rules: a set of special rules that are not used in volition formulation and are not associated with any specific instantiation. If certain conditions hold true, they update the social state.

The four core components of the CiF architecture are: **social state**, **characters**, **social exchanges** and **trigger rules**.

The **characters** use their perceptions to acquire information about the **social state** of the world. In order to fulfill their desires, they try to change the **social state** to accommodate them through the use of **social exchanges**. These exchanges are social interactions that can end positively or negatively for the **character** that started it, the initiator, but will necessarily have consequences, depending on the receiver's response.

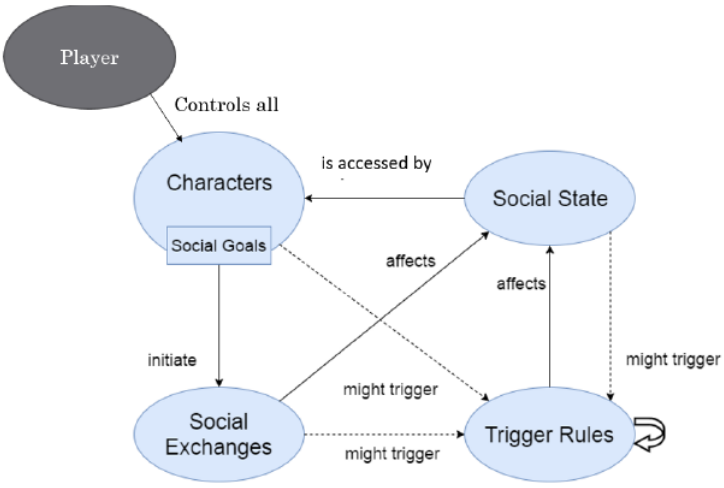


Figure 2.4: Interaction between core CiF components [5]

These consequences directly affect the **social state** on which the **social exchange** was performed. As a reaction to the changes of the social world, **trigger rules** can be activated and initiate a cascading effect resulting in a dramatic change of the **social state**.

2.2.1.B CiF Algorithm

- **Desire Formation:** In the first step, the initiator's volition to start a social exchange with the other characters is computed by counting the weight of the rules in both microtheories and the social exchange's initiator IRS. This process is done for every character in the world.

- **Intent Formation:** Once this is done, the exchange that is most desirable is chosen and created with the initiator's intent, the initiator and the receiver.

- **Perform Social Exchange:** The receiver ascertains if they will accept or reject the social move, depending on their own rules, similarly to the first step, Desire Formation.

- **Performance Realization:** Then the system will select the most salient effect to enact, the one with the most profound changes. The effect is asserted and the instantiation is instanced.

- **Social Fall Out:** After the outcome of the social exchange is established, the new social state, the consequences of the interaction are simulated by running a set of *trigger rules* that take into account the changes made in the new social state, updating the character's status in the SFDB, which makes reasoning over these consequences possible in the future social exchanges.

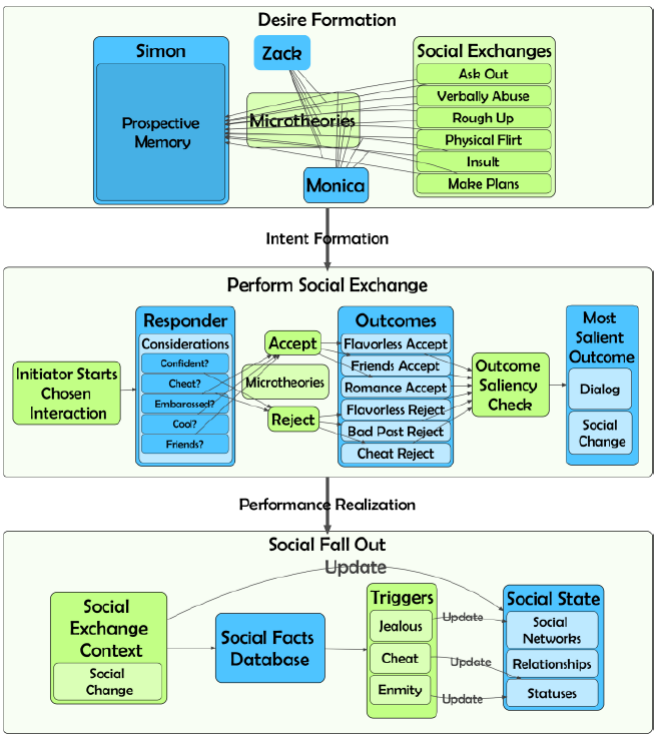


Figure 2.5: CiF algorithm [2]

2.2.2 *Ensemble*

Ensemble [14] is the latest version of the CiF architecture. Ensemble innovates by presenting instantiations that are not fixed, but whose path is generated on the go, instead of using static branching tree structures or state machines. These playable social dialogues are called *social practices*. The selection of the character's response takes into consideration not only the structure of the social exchange happening but also the social-cultural norms of the world.

In the previous iterations of the architecture, instantiations were scripted dialogues (accompanied by character animations) that once set on motion would play by themselves until the end. Ensemble adds another layer of interactivity by turning instantiations into dialogues where the player and the NPC take turns, choosing what to say at each stage.

When video games implement dialogue-trees to grant the player some freedom in the dialogues it is common to observe moments of "amnesia" in the NPCs lines. If the player sets in motion a dialogue-tree and makes a choice that regrets or just wants to explore another route, when given the choice they can restart it and choose some other option while the NPC seems completely oblivious of the talk that they just had in that same session. Ensemble reasoning maintains the consistency and coherence of the conversation as it progresses.

Below are the definitions of some new concepts introduced with Ensemble.

Social Practices: interactive dialogues between characters. Instead of using a static dialogue-tree structure, the social practice is not strongly pre-authored but generated as each of the involved parties chooses their action. The selection mechanism uses previous actions and the current social state to setup the next stage of the dialogue, ensuring consistency. Social practices are made-up of linked *stages*, each with a selection of *actions*, made by the characters. The path through a social practice is defined by scoring actions (based on the social state) and microtheories. This process ends when there are no further linked stages to the current stage.

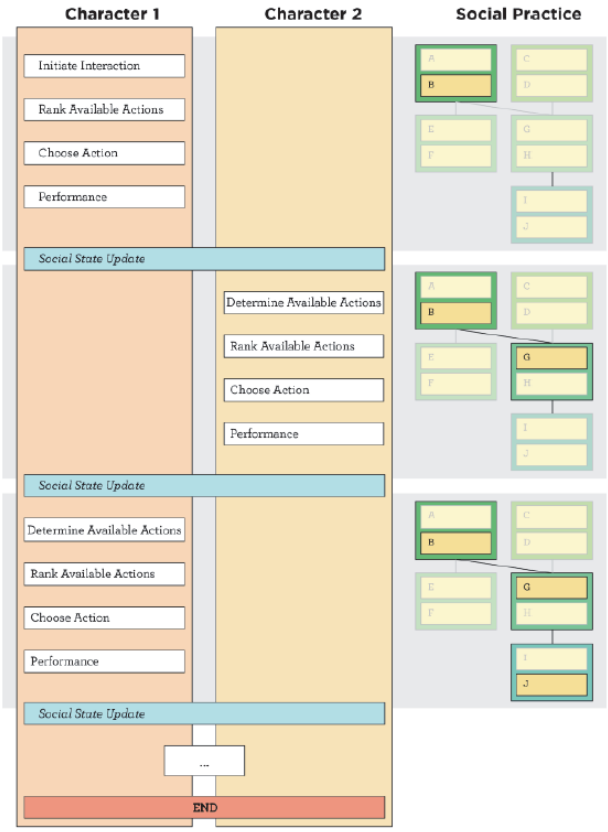


Figure 2.6: Diagram of a social practice. Characters A and B take turns selecting actions that move the social practice to a new stage. Actions are represented by letters from A to J. In this figure, each block of two actions is a stage. [14]

Stages: sets of actions that offer alternatives to represent the same aspect of a practice (e.g. in the stage ”reciprocate compliment” the actions should be different ways to reciprocate a compliment). Stages are linked under the constraints of a directed acyclic graph (DAG).

Event Stages: special stages that are triggered depending on the character's circumstances. If two characters hate each other and are interacting, an event stage where one of the parties refuses to talk to the other and stops the social practice early can be authored.

Actions: express an agent's intent. When an action is chosen, a performance consisting in dialogue

combined with visual feedback is played and the social state is updated via the action's effect.

Action Scoring: all of the actions in the linked stages are considered and the one with the highest score is chosen. The score is a sum of three components: the microtheories associated with the action's intent, the action's influence rules and the *practice-specific bias*. The first two are similar to the desire formation in the original CiF. The practice-specific bias is a weight that, independently of the social relationship between the characters, expresses the most routine path through a practice. For example, most people answer "How're you doing?" with "Good! How're you doing?".

Social practices end up lessening the burden of authoring on the developer. The chosen representation, the DAG, provides a structure to help with visualization. The most used structure in nowadays video games is the behaviour tree, it suffers however from poor scaling when adding complexity, growing exponentially. This happens because the tree has many dependencies on every level that create hard constraints, so every combination (path) must be explicitly represented. In contrast, the paths on the DAG have soft constraints (influence rules, microtheories, practice-specific bias), therefore the impact of dependencies on the complexity is lessened, allowing the author to add, remove and modify authored content with minimal effort.

Project Perdido [6] (working title) is a visual novel video game that uses Ensemble and is being developed by the authors of the architecture.

2.2.3 *Comme il Faut - Creation Kit (CiF-CK)*

CiF-CK [3] is an implementation of CiF that has Role-playing Game (RPG) as focus. The term "quest" is very familiar to any RPG: a plot driven mission that has initial conditions, an objective and effects. CiF-CK uses the parallels between the notion of *quest* and CiF's *social exchanges* to create an adapted model than can be generalized for any RPG.

Diverging from the original CiF, in which every character has its personality defined by the system and can be controlled by the user, CiF-CK puts the player on the role of a single character and allows them to role-play and shape that character's story.

A notable addition to the CiF model is the notion of *belief*. A character may believe that one fact is the truth and act accordingly, despite possibly being wrong. The NPCs' desire formation now takes into account not only their goals but also what they believe to be true.

CiF-CK was implemented in the video game *"The Elder Scrolls V: Skyrim"* [12], a role playing game (RPG) developed by Bethesda Game Studios and published by Bethesda Softworks. This system is easily extended to RPGs developed with Bethesda's engine, the Creation Engine, such as Fallout 4 [15]. The resulting work not only accomplished its goals of creating a better gaming experience with interesting and believable characters, but also resulted in a mod that achieved success within the player

[6] http://www.asdesigned.com/creative/perdido/

Table 2.1: Relation between CiF and CiF-CK Characters Social Exchanges [3]

CiF	Function	CiF-CK
Name	Unique Identifier	Quest ID
Intent	Intended Social State Change	Used once the Quest is finished
Pre Conditions	Conditions for it to happen	Quest Start Conditions
Initiator Influence Rules	Character's Desire to initiate the exchange	Set of Rules
Responder Influence Rules	Target's Reaction to the Exchange	Set of Rules
Effects	The resulting consequences of the Exchange	Applied in the Final Stage of a Quest
Instantiations	Performance of the Social Exchange	Dialogue between NPCs

community and the media [7].

CiF-CK takes advantage of the Creation Engine's quest system and uses the said of the quests to the social exchanges when representing the later. This allows CiF-CK to create playable content that is custom to the player's playthrough that guides the narrative organically.

The mod was implemented resorting to the *Creation Kit*, a tool supplied by Bethesda that allows adding and modifying content and was used to develop the game. The Creation Kit provided the structures necessary to create a version of the CiF architecture.

A – Social Exchanges Skyrim is a RPG, a video game genre that traditional has a big emphasis on questing. As mentioned before, the Creation Kit quests and CiF social exchanges have structures that can be matched, allowing the social exchanges to be adapted as quests.

B – Characters The characters in CiF and CiF-CK are nearly the same in structure, with the adaptation that in CiF-CK desires are expressed by quests.

C – Social State Social States in CiF-CK show one major difference from the CiF Social States, the Social Networks are private. This is what enables the belief system to exist, as the characters stop being omniscient and the values in their Social Networks become only a belief, not the absolute truth.

A technique that video game engines use to reduce the resources that need to be allocated is to compute information of the neighbourhood around the player and ignore the rest. Skyrim is no exception to this optimization. The problem is that because of this NPCs that are far away from the player would

[7] https://www.nexusmods.com/skyrim/mods/77792/

Table 2.2: Relation between CiF and CiF-CK Characters [3]

CiF	Function	CiF-CK
Name	Unique Identifier	Unique Identifier
Gender	Describe the Character's Gender	Permanent Value
Traits	Permanent Traits	Permanent Values that affect social exchanges
Status	Temporary Traits	Temporary values that affect social exchanges
Prospective Memory	Set of desires of social exchanges with specific goals	Set of quests with specific actors (Targets)

Table 2.3: Relation between CiF and CiF-CK Social State [3]

CiF	Function	CiF-CK
Social Networks (public)	Feelings characters have for each other	Unique for each Character (private)
Relationships	Publicly recognized relationships	Public information is stored in a static reference and accessible to all
Cultural Knowledge Base (CKB)	What each NPC likes	Unique and different for each character
Social Facts Knowledge Base (SFKB)	Social History of the World	Public information is stored in a static reference and accessible to all

not be considered. The design choice to have relationships as a public static entity is the solution to this problem. This way the system has always a reference to any NPC.

D – Trigger Rules

Trigger Rules in CiF-CK work just like in CiF. After a Quest (Social Exchange) is finished (and the instantiation is asserted), CiF-CK runs all trigger rules and applies the eventual changes to the Social State.

E – CiF-CK Implementation

The system was implemented using 3 main scripts:

- **CiF-CK script** that manages the NPCs' decision making cycle;
- **Influence Rule script**, an auxiliary script that computes the volitions for the social exchanges to be performed;

- **GameManager script** that manages the social state of the location that the player is in. From the surrounding NPCs, decides which one is performing a social interaction. It also notifies the other NPCs of any social state change, allowing them to update accordingly.

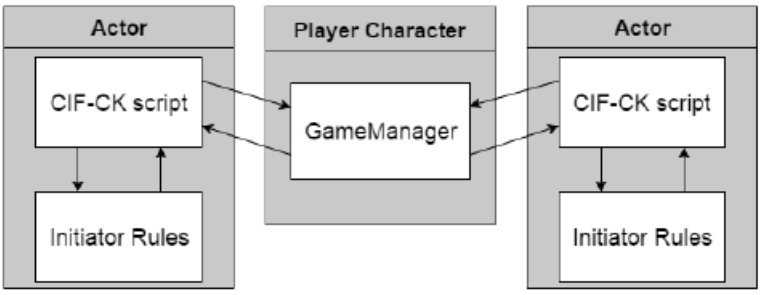


Figure 2.7: The internal structure of CiF-CK [3]

2.2.4 *Fearnot Affective Mind Architecture (FAtiMA)*

FAtiMA [6] is an agent architecture that endows agents with emotions and personality that, joined with the architecture's planning capabilities, affect their behaviour. FAtiMA uses **emotions and emotional behaviour** based on the OCC Model of Emotion [16] to achieve believable characters that evoke empathy in the user. The OCC Model of Emotion is a conceptual model that characterizes 22 different emotions and organizes them in a hierarchy that explicits the conditions for them to be active.

A modular version of FAtiMA was developed [17]. Thanks to the modular architecture, only the necessary components for each project need to be present, making the system lighter. The core architecture can be augmented with components like the *Theory of Mind component* that creates a model of the internal states of agents or the *deliberative component* that handles goal-based behaviour and adds planning capabilities.

Each *character* perceives the environment using their *sensors*. The agent appraises the perception's significance and triggers the appropriate *emotions*. If a *goal* becomes active an *intention* to achieve that goal is created. The resulting *emotional state* is sent into *action-selection* for two different levels: *action-tendencies* (more instinct reactions like getting angry when you are attacked by another wastelander) and *coping behaviour* (reactions that have some reasoning behind them, resulting from the internal goals of the agent, like asking the wastelander to stop).

Coping behaviour can be divided into two kinds of coping: *problem-focused coping*, involving planning and acting to achieve goals, and *emotion-focused coping*, in which the character's interpretation of the environment is altered. If the aggressive wastelander does not stop the attack, the character may

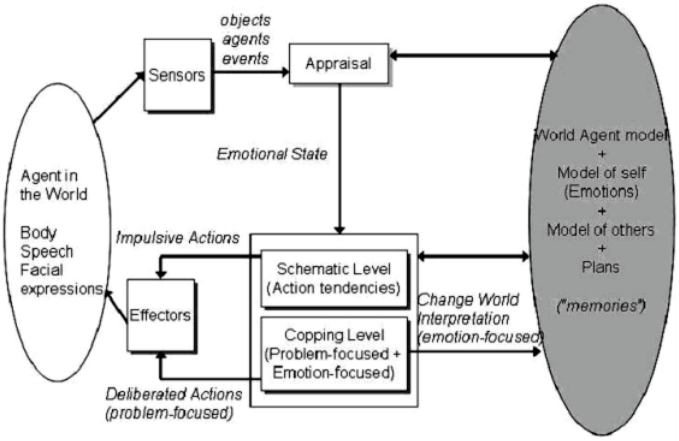


Figure 2.8: FAtiMA's simplified architecture [6]

feel distress by failing to complete its goal and lower this goal's importance, effectively reducing the distress caused by the failure. As illustrated, emotions can not only influence the agent's reactive behaviour but also the planning process. After this, the character acts on the world using its *effectors*.

2.2.5 *GAMYGDALA*

GAMYGDALA [18] provides NPCs with OCC emotions. Game events affect NPC Goals, which in turn react by producing specific emotions to the NPC. This way, different NPCs can generate different emotions when submitted to the same situation.

The author defines the relationships between the NPCs, attributes a set of goals to each and how the game events impact these goals (how good or bad a game event is to the goal). The system's emotional appraisal process is notified when events occur, which in turn outputs a generated emotion. GAMYGDALA works like a black-box module that operates totally independently to the NPC's AI, so the developer does not need to be an expert in emotional appraisal when integrating it on a system.

2.2.6 *PsychSim*

PsychSim [7] is a system that allows agents to have social interactions. The user can use PsychSim as a tool to define scenarios with individuals and groups, each with their own goals, relationships, beliefs and mental models and watch how they interact and influence each other.

Agents in this system have fully specified decision models of other agents. This illustrates how in real life human actions are made taking into account predictions based on mental models that we have of other people. These predictions can be referred as *"theory of mind"* [19].

PsychSim's usage can be divided into three moments:

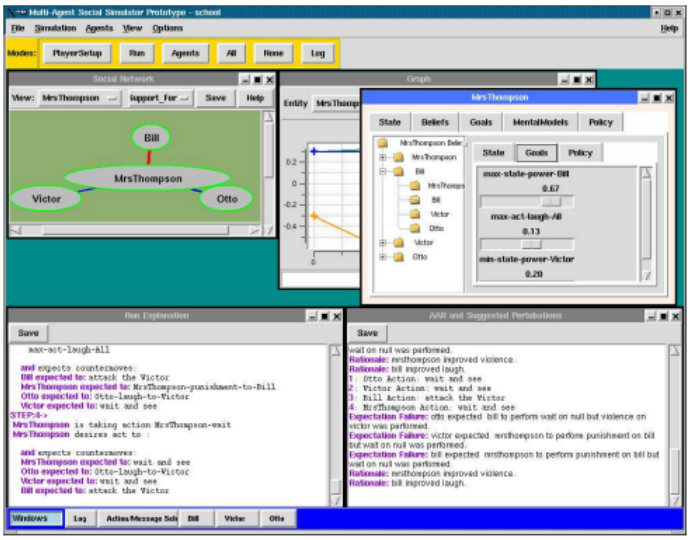


Figure 2.9: PsychSim's interface [7]

- a phase that the user does the environment setup, populating the world with entities, their models and behaviours;

- the simulation phase, where the user can control what happens and watch as the simulation unfolds;

- the analysis/perturbation phase that supports an iterative refinement of the simulation. The results of the simulation are stored in a database that will be accessed by inference rules and explain the reasoning behind the agents' results.

2.2.7 *Synthetic Group Dynamics (SGD)*

SGD [20] is a model that meets the user's expectations by following believable group dynamics between virtual agents, inspired by theories developed in human social psychology sciences. The different types of interactions that may occur in the group alter the dynamics, producing a positive effect on the user's social engagement.

The group is composed of various synthetic agents that interact with each other. Agents are aware of the other individual members and of the group itself, being able to build proper knowledge regarding the group's social structure, using this information to motivate their behaviour. The interactions impact the group state and, at the same time, are influenced by that state. SGD influences the perception, knowledge building, behaviour and action process of each agent.

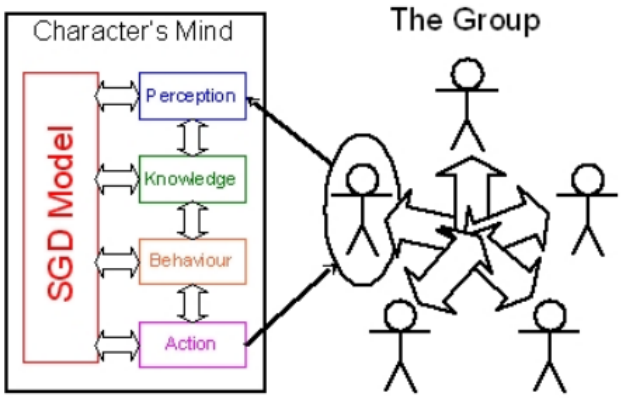


Figure 2.10: The SGD model in the mind of each member of the group [8]

2.3 State of the Art

In this section we present two projects that enhance Conan Exile NPCs' behaviour as well as two video games that notably explore social interaction with agents.

2.3.1 A Merchant Model

A Merchant Model [9] set the objectives of delivering an improved experience for Conan Exiles by doting the game with merchants whose range of interactions is vast and believable while maintaining the authoring burden low. Usually video games have merchants whose purpose is to solely exchange products, lacking in social capabilities.

The resulting merchants have businessman abilities (they try to make profit by making several deals to the player) but also feel human, being able to form a *relationship* with the player which is reflected not only on the prices that the merchant has to offer but also on the dialogue with the player. The merchant has *traits* that affect how easily the player can get on its good side and the tone of his lines (*general traits*) and the kind of deals that the player will get (*merchant traits*).

The trader's memory is made of three components: *social fact database*, *social networks* (both have the same function as the CiF architecture's counterpart) and *player preference database*, whose purpose is to save a list of items and actions that the merchant knows the player likes. The player preference database is populated when the merchant has contact with the player and learns more about them.

The merchant also has a cultural knowledge database that much like CiF-CK expresses his preferences on objects.

The dialogue system is not native to the game and had to be implemented for this work.

User tests were conducted using two scenarios: one where the merchant was a basic one, without social capabilities, and one where the merchant displayed emotions (with a nice or arrogant personality). It was concluded that the experience was a success: players prefer to interact with a merchant with a

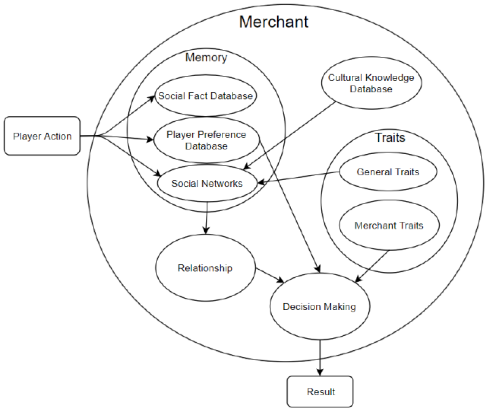


Figure 2.11: The merchant's architecture [9]

social component embedded (even if the merchant is arrogant).

2.3.2 A Procedural quest generator for Conan Exiles

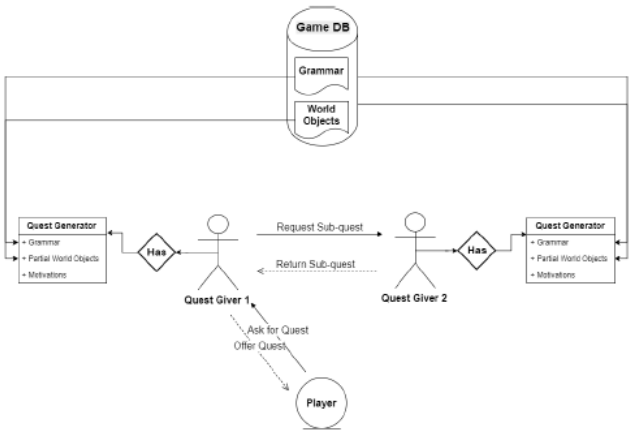


Figure 2.12: The quest system architecture [10]

A Procedural quest generator for Conan Exiles [10] has the objective of creating a model that procedurally generates and monitors the completion of quests in Conan Exiles, a video game that does not have a native quest system.

A new type of NPC was created using a component that generates quests. These quest giver NPCs create quests that satisfy a motivation (conquest, wealth) and a strategy (attack an enemy, gather raw materials). The quest will borrow elements like characters, items and locations from the Conan Exiles' world database. NPCs are also capable of communicating with other NPCs in order to generate sub-quests with their own motivations.

User tests were conducted using two scenarios: one where the player would play the vanilla Conan Exiles experience and other where the player would find quest givers in the world. The user tests

concluded that the quest system did not worsen the overall experience for the players.

2.3.3 Façade

Façade [21] is a video game that takes the player out of his comfort zone, placing them in the apartment of a couple, Trip and Grace, giving them little to no context on his role as they watch a domestic confrontation unfold. The player can intervene (and is asked to do so by the couple), giving his opinion by typing on the keyboard. The game uses natural language processing to give liberty of choice on how to solve the dispute or even escalate it.

Figure 2.13: Trip and Grace [8]

Despite being a small game, Façade uses a lot of custom experimental technology, rarely seen in video games, even twelve years later, such as: *A Behavior Language (ABL)*, a special purpose programming language designed for authoring personality rich, emotive autonomous characters, used to program the two agents; a rule language for authoring robust natural language understanding rules, *NLU*; a discourse management framework for keeping track of multiple conversational threads and deciding which response to give in the current context.

Façade is a charming experience that conjugates narrative, interactivity and procedural simulation, a landmark of social AI interaction.

2.3.4 The Sims

The Sims [22] is a series of video games that has a social agent simulation model at its core. The NPC characters have personality traits, desires, mood, needs (both social and physical), relationships, an extensive number of possible social interactions (compared to another video games) and, in The Sims

[8] http://www.interactivestory.net/

4, emotions. The player is able to control multiple NPCs belonging to a family. If a NPC has strong desires or urgent needs they are able to bypass the player's orders and reorder their action queue.

As there is no "end game" and no objectives defined, the social exploration and development of the world is the main focus of the gameplay. Social interactions help shape the world, impacting the participating NPCs' desires, mood, emotions and social standing with other observing NPCs.

These interactions can be "Friendly" interactions (like *Talk*, *Hug* or Ask about day), "Romantic" interactions (like *Flirt*, *Kiss* or Ask if single) and "Mean" interactions (like *Insult*, *Yell* or Slap). Some interactions can be more complex and use the game's social structures and events, like *Mock (trait)*, *Ask to join (existing activity)* or *Break Up*.

Figure 2.14: The Sims 4 screenshot [9]

The Sims series offers a highly dynamic social environment and NPCs with rich social capabilities for players to toy with. Its commercial and cultural success are a testament of the value that social AI can be to a video game.

[9] http://quovx4d83tr2hp1r22mgwa1m-wpengine.netdna-ssl.com/wp-content/uploads/2014/09/sims4interactionchoices.jpg

2.4 Discussion

Conan Exiles is a AAA video game that features a rich world environment populated by various NPCs (some of which possess basic social capabilities) and modding tools, presenting itself as a great candidate to be the testbed for our model.

We will now consider the main models mentioned: CiF, FAtiMA, GAMYGDALA, PsychSim and SGD. We have to acknowledge what we want to accomplish with this work: improving the *User Experience* and the NPC Believability in Conan Exiles by implementing a social model.

GAMYGDALA's objective is to simulate emotions, as such reasoning, planning, decision making and most importantly, the social interaction's representation are not included in the model. By its own it would only solve part of our problem. SGD presents an interesting approach to social interaction, using group dynamics to create believable agents that build their knowledge using the past group interactions and their dynamics. Like GAMYGDALA, SGD focuses on solving a very specific problem and would require further work.

Both FAtiMA and PsychSim are very complete systems. Performance is however one of the concerns when developing video games. Despite wanting a model that displays good results, we have to make sure that the performance of the game is not affected. We fear that such could be the case with FAtiMA and PsychSim. CiF-CK has however been implemented on a commercial game before and proved to assure the stability of the gameplay.

CiF models NPCs with traits, status, relationships and social networks between them, and an algorithm that appraises the NPCs' desires, starting social exchanges that will modify the social panorama when concluded.

The poor scalability of the authoring load is a relevant problem when creating a social model. As a test environment, it is to be expected that our implementation will have only a handful of NPCs, but if we were to author for all the NPCs in a AAA video game, we would have to dispense countless hours. It is essential that the chosen social model is designed with this complication in mind. One of the major objectives that the CiF architecture aims to solve is the exact above.

As such, CiF is an ideal suitor to become the base for our solution. FAtiMA and GAMYGDALA explore and model emotions, an interesting concept that we will approach and expand upon.

CiF-CK is an extension of the original CiF model, tailored to fit the needs of the platform that it was implemented on: the RPG video game *The Elder Scrolls V: Skyrim*. The modifications that CiF-CK proposed such as the the Player controlling only one character (highlighting the role-playing aspect of the game) and the addition of a belief system that opens up new social exchange possibilities, have proven their effectiveness in creating a rich interactive social AI environment. We expect that our system, deeply inspired by the above mentioned, will also achieve satisfactory results.

3

Comme il Faut - Exiles

Contents

3.1 Solution 28

3.2 CiF Cycle 36

In this chapter we will detail our model, the **Comme il Faut - Exiles (CiF-Ex)**. This model is based on the Comme il Faut [2] architecture and takes inspiration on the concepts introduced in the Comme il Faut - Creation Kit architecture [3].

Our model extends the CiF and CiF-CK models by:

- **exploring character emotions (in the form of *Emotion and Social Emotion Status***, based on the OCC model of emotion [16]). The CiF-Ex emotions fluctuate as an immediate consequence of a Social Exchange and **contribute to create detailed characters that respond realistically**;
- **extending the CiF-CK's *Social Network Belief***. Instead of just keeping track of the Social Network values that the NPC believes the other NPCs have towards them, NPCs also have beliefs on how much the other NPCs like each other. This allows us to **convey complex behaviours that rely on the expectation that NPCs have for the others' relationships**. For instance, a NPC can notice that their loved one has a great affection for another NPC and feel *envy* for the latter, expressing it by desiring to initiate negative social interactions with them;
- **extending the representation of the *Status* to a continuous scale** instead of a binary variable. This way we can express the intensity of an emotion: rather than just declaring that a NPC does or not feel an emotion, we can **detail the depth of that emotion and also decay its value with time**;
- **adding a new possible Social Exchange Outcome, *Neutral***, to the already existing *Positive* and *Negative*. Having two polar possible reactions to a social exchange is a simplification of what happens in the real world, people do not just answer intensively to every social interactions. The two outcomes also limit the author's freedom of expression for the dialogue, having a middle ground allows them to express two opposites and also a more calm response;
- **giving the Player the ability to respond to all the Social Exchanges** that the NPCs initiate with them, determining the Outcome. Instead of being a passive receiver, **the Player can role-play their character and articulate their feelings, deciding the outcome of the Social Exchange**. CiF-CK already allowed the response to some special SEs (like "Dating proposal").

3.1 Solution

The CiF-Ex architecture features five main components:

- **Social Exchanges**: the space of possible social interactions;
- **Characters**: a structure that holds the social attributes of an NPC;

- **Social State**: a structure that represents the social panorama;
- **Trigger Rules**: sets of rules that when activated change the Social State.
- **Microtheories**: General rules applied to the initiator and the receiver for all the SEs.

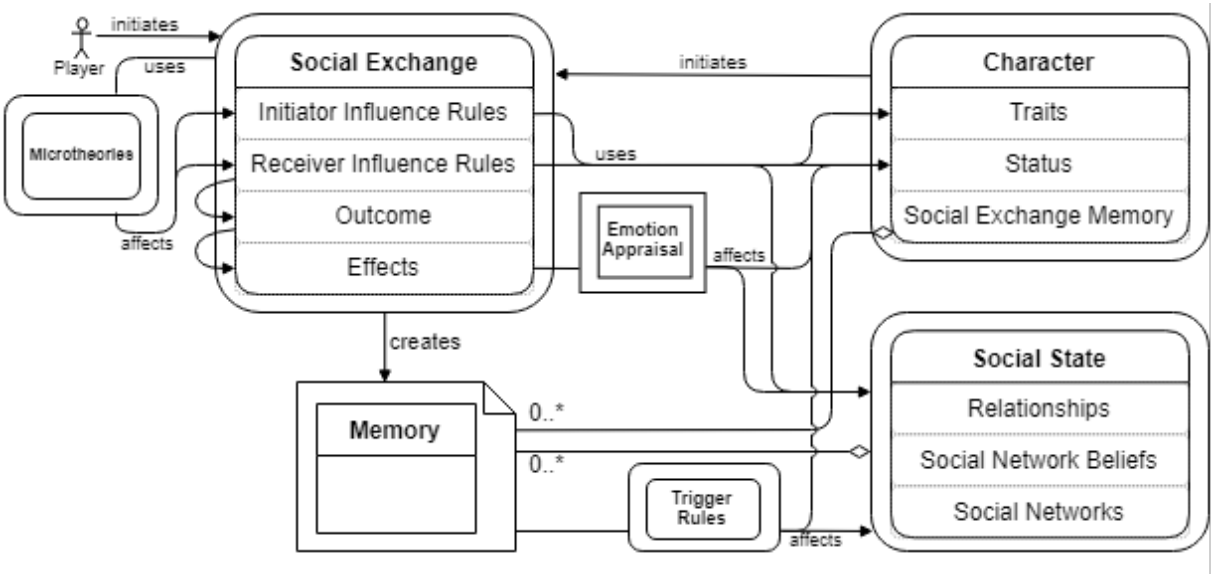


Figure 3.1: The CiF-Ex Components

The Figure 3.1 shows the CiF-Ex components and their interaction. **Characters** initiate **Social Exchanges** that they desire in the role of the *initiator*, setting another character with the role of the *receiver* (some SEs may even have a third character participating, the *other*). The Social Exchange's *Influence Rules* use the characters' **attributes** (*gender, traits, status and social exchange memory*), **Social State** and **Microtheories** to calculate the volitions, quantifiable representations of the NPCs' desire to carry out a specific SE.

The receiver's volition determines the *Outcome* of the Social Interaction, if it ends positively, neutrally or negatively. In turn, the Outcome dictates both the *Effects* (the immediate changes to the characters' Status (*Emotion Appraisal*) and Social State) and the *Instantiations* (the character's visual performance of the Social Exchange). The end of a SE is marked by the creation of a **Memory**, an object that summarizes a past SE, including some metadata. The new memory is archived by the participating and other characters that witnessed the SE in their *Social Exchange Memory* and by the Social State's *Social Facts Knowledge Base*. The **Trigger Rules** process the new memory for each character that receives it, affecting their Status and the Social State. Now we will define in greater detail each CiF-Ex component.

3.1.1 Social Exchanges

Social Exchanges are structures that represent the social interactions (in this work we focus mainly on dialogue interactions) that the characters and the player can have with each other. The outcome of one SE is dependent on the Social State, personality traits and status of the SE participators and has the capacity of modifying the Social State.
Compliment, *Insult* and *Brag* are some examples of SEs. It is up to the author to decide whether the SEs that they craft are of a more universal significance or more specific, depending on the detail that they want to reproduce (e.g.: having one *Insult* SE vs having one *GraveInsult* and one *LightInsult* SEs with appropriated social impact and dialogue lines). Social Exchanges are composed by:

- **Intention** represents the purpose behind the SE (the *Insult* SE has *Negative* as intention, the *Compliment* SE has *Positive* intention, the *Flirt* SE has *Romantic* intention). Intentions are used to generalize behaviour when calculating Microtheories and Trigger Rules.

- **Influence Rules** are used to calculate *volitions*. The **Initiator Influence Rules** are used to prioritize which SE the initiator wants to perform the most and the **Receiver Influence Rules** are used to calculate how favourably the receiver will react to the SE. The parameters used by Influence Rules are traits, social network and status values.

  The Influence Rule is a weighted sum between a vector of the influence rule Weights (different for each SE) and the actual Values for each parameter:

  v_i is the current value of the parameter

  w_i is the weight of the parameter

  $$Volition = \sum_i v_i w_i \tag{3.1}$$

  For example, if there are three status in total, *Happiness*, *Disgust* and *Hunger*, with the values [4, 0, 1]and the Influence Rule Weights are the vector [3, -1, 0]the resulting volition would be equal to: $4 * 3 + 0 * (-1) + 1 * 0 = 12$.

  The only exception to this rule is the *Goal* value of the Social Networks (see Section 3.1.3). Instead of using the direct value, we subtract the *Current* value to the *Goal* value. This means that if the Goal is greater than the Current value the character will desire to perform SE that improve the Current value. For instance, if the *Friendship* social network's Goal is 5 and the Current value is 3, the value used in the Influence Rule will be: $5 - 3 = 2$.

- **Pre-conditions** are rules that are calculated before the Initiator Influence Rules (and only influence the initiator volition). These rules disable SEs that would not be coherent with the current social context (for example, a NPC should only introduce itself once to the Player).

- **Outcome** can be *Positive*, *Neutral* or *Negative*, depending on the calculated receiver volition and will dictate the reaction of the receiver. Each SE has two different thresholds that decide the Outcome: if the volition is less than the lower threshold value the outcome is *Negative*, if it's in-between the values the outcome is *Neutral* and if it's greater than the higher value, the outcome is *Positive*. The outcome is used to select what instantiation will be acted and Effects of the SE.

- **Effects** are the consequences of the SE. Effects modify the participating characters' Status (see Table B.3 for more information), their Social Networks, Beliefs and Relationships (see Table 3.4 for more information). For each SE and possible Outcome there are fixed values that are added to the *Social State* and *Character Status*.

- **Instantiations** is a term originated from the CiF architecture. These are the "theatrical performances" of the SEs' outcomes. *Instantiations* are mainly dialogue lines specific to each SE participator complemented by NPC animations.

The Table B.1 briefly describes the internal structure of a SE. A list of the implemented Social Exchanges can be seen in Table B.2.

3.1.2 Characters

A **Character** is a structure that holds the CiF data necessary for the NPCs' social behaviour. See Table B.3 for a short description of the Character elements.

- **On cooldown** is used to verify if a character is on cooldown. After performing a SE the characters go into cooldown and aren't able to start SEs until this expires. The time of the cooldown is dependent on the character's traits (e.g.: a character with the trait *Shy* will have a greater cooldown than average).

- **Busy** is used to check if other characters can start interactions with this character. If a character is targeted as the receiver for one SE (either by another character or the Player) they will be flagged as *busy*. The Player is able to interrupt an on-going SE, bypassing the busy flag.

- **Traits** are used to give the characters a distinct conduct, they shape the character's personality. Traits are predetermined when the character is authored and immutable. *Friendly*, *Hostile* and *Brave* are examples of traits. The list of Traits and their descriptions can be seen in Table B.4.

- **Status** are volatile attributes that portray the character's emotional condition. Status are changed at the end of all SEs and decay with the passing of time. Status have three categories: **Emotion Status** (depicts the character's mood, e.g: *Anger*, *Happiness*) , **Social Emotion Status** (depicts

what the character is feeling towards other characters, e.g: *Admiration*, *Resentment*) and **Physical Status** (depicts the character's physical needs, e.g: *Hunger*, *Tiredness*).

The Status in our architecture are greatly inspired by the OCC Model of Emotion [16]. A list of the Social Emotion, Emotion and Physical status can be seen in Table 3.1, Table 3.2 and Table 3.3, respectively. The descriptions for the Emotion and Social Emotion Status are based on the descriptions of the OCC Model.

Table 3.1: Examples of the Social Emotion Status

Status name	**Description**
Shame	disapproving of one's own blameworthy action
Admiration	approving of someone else's praiseworthy action
Reproach	disapproving of someone else's blameworthy action
Pity	displeased about an event presumed to be undesirable for someone else
Gloating	pleased about an event presumed to be undesirable for someone else
HappyFor	pleased about an event presumed to be desirable for someone else
Resentment	displeased about an event presumed to be desirable for someone else
Anger	disapproving of someone else's blameworthy action

Table 3.2: Examples of the Emotion Status

Status name	**Description**
Sadness	displeased about an undesirable event
Happiness	pleased about a desirable event
Disgust	disliking an unfamiliar aspect of a character
Fear	displeased about the prospect of an undesirable event
Satisfaction	pleased about the confirmation of the prospect of an undesirable event

The process of adjusting the *Emotion and Social Emotion Status* is called the **Emotion Appraisal**. When a Social Exchange is performed, the Emotion Appraisal uses the asserted Outcome to apply a set of changes to the participants' Emotion and Social Emotion Status. As such, there is one set for each Social Exchange and possible Outcome.

Both CiF and CiF-CK represented status as binary, a character was under a certain status or not. In our model, each status is represented by a value between 0 and 10, which allows for a greater detail when describing a character's disposition. The influence rules take advantage of this: instead of just checking if a character is *Angry*, we are able to check how *Angry* the character

Table 3.3: Examples of the Physical Status

Status name	**Description**
Hunger	Influences the Ask For Food and Offer Food SE volitions. Has a probability of increasing by itself
Thrist	Influences the Ask For Drink and Offer Drink SE volitions. Has a probability of increasing by itself
Tiredness	Increases periodically and each time a character starts an SE as the initiator. When maxed out, the character will go sit by their home and rest

is and create rules that take this distinction into account. These values are changed at the end of SEs, depending on the Outcome.

Status can also gradually decay (for example, characters can progressively get less angry) or increase (for example, if you are modelling a Physical Status like hunger) with the passing of time.

- **Social Exchange Memory** stores representations of the SEs that happened in the vicinity of the character, including the ones that did not involve them. Each individual memory (one entry of the SE Memory) is composed of the names of the participators, the SE's name, the outcome, the origin of the memory (it will be the character's name if they witnessed the SE themselves), the world's timestamp of the event and an ID that is generated to identify each SE. The characters are able to look-up their memories which will in turn influence volition calculations. A memory is added to the SE Memory if the character is within a fixed distance of the SE initiator when the SE is happening.

The user is an entity that interacts with the CiF components. In both the video games that CiF-CK and CiF-Ex were implemented in (Skyrim and Conan Exiles, respectively) the user impersonates his own character. However, in the original CiF architecture the user was able to control all the characters. Conan Exiles would not be the most suitable video game if we were to keep this design choice as we would be required to dramatically change its base gameplay. As such, in CiF-Ex the user only controls one character, which will be referred as the "**Player**". This change was proposed in CiF-CK and was kept for our architecture.

The user is able to choose any SE that they wish to perform using the Player as a proxy. The trigger rules are not calculated for the Player since they have no internal attributes to be modified. From the perspective of the CiF-Ex components the Player is just like another character; characters will want to interact with the Player, the Social State will keep the values related to the Player and the SEs can be initiated by the Player just like a NPC would.

Considering that in the CiF architecture the NPCs were controlled by the Player, the NPCs did not have a decision making cycle that chose what SE they were going to perform next, they just stood around, so the variables *On Cooldown* and *Busy* that aid managing the NPCs' actions were not present.

Table 3.4: Social State in CiF-Ex

Term	Function
Social Networks	Non-reciprocal feelings towards other characters
Social Network Beliefs	Beliefs about the other NPCs' social networks
Relationships	Reciprocal social connections
Social Facts Knowledge Base	Records the social interactions of the world

3.1.2.A NPC Cycle

The NPC cycle begins by the NPC's default behaviour, *Idle*. At any time, if the NPC is nearby an ending Social Exchange performance, they can *receive a SE memory and process* it. If the NPC is selected as initiator, they can start a SE with the receiver. If the Player interacts with a NPC, the NPC participates in the SE as the receiver.

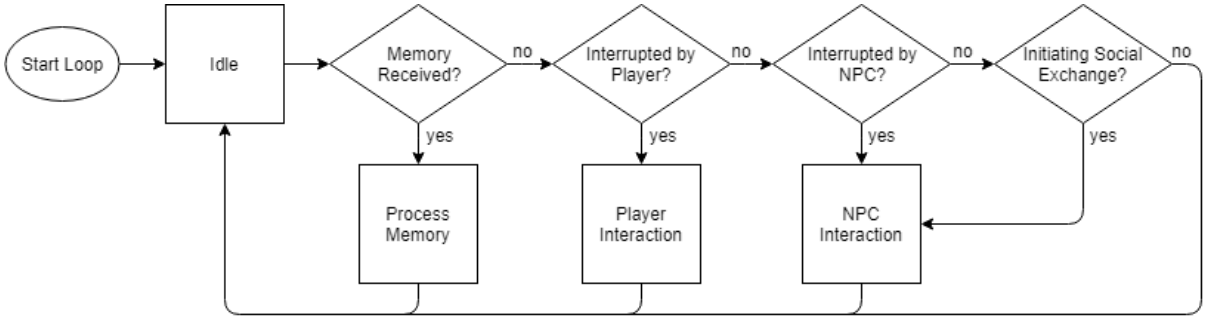


Figure 3.2: CiF-Ex NPC Cycle

3.1.3 Social State

The **Social State** is the snapshot of the social panorama at any time. This tracks the evolution of the social scene of the world. By examining the *Social State* the social dynamic between the NPCs (and the Player) are evident: *Social Networks* represent how NPCs feel about each other, *Relationships* are strong associations between them and the *Social Facts Knowledge Base* holds all the SEs that have occurred.

- **Social Networks** are private feelings, that may or may not be reciprocal, felt by character towards another. Each Social Network has three variables: *Current* (the current value of the network), *Goal* (the value that the character wants the Current value to tend to) and *Belief* (the current value that one character thinks the other character has for the former, as introduced by CiF-CK). As a result of being non-reciprocal, the Belief value can be wrong. Examples of Social Networks are the *Friendship* and *Attraction* networks.

- **Social Network Beliefs** works the same way as the Belief variable of a Social Network but instead of connecting themselves with one other character, it connects other characters ("Lydia" thinks that "Benor" has a negative *Friendship* value for "Farkas"). There is one Social Network Belief structure for each Social Network that exists.
- **Relationships** are reciprocal social relationships like being *Friends* or *Dating*. Relationships are private, so two characters can be in a relationship without the others knowing about it. Relationships can be inferred using the Social Network Beliefs.
- **Social Facts Knowledge Base** is an archive of all the SEs that happened in the world. Much like the SE Memory (see Table B.3) it saves memory entries.

3.1.4 Microtheories

Microtheories (MT) are rules applied when calculating the pre-conditions and the initiator and receiver volitions. These rules represent conventional behaviour in the world, social norms. As an example: if "Lydia" and "Benor" are friends, "Lydia" is expected to desire SEs with the *Positive* intention and "Benor" is more prone to positively accept them. As Microtheories represent general attitudes that apply to all the Social Exchanges, the author won't have to detail every nuance when defining the SEs influence rules, effectively reduce the authoring burden.

Examples of Microtheories are:

- *Dating MT:* if the characters are dating, they will desire to perform SEs with the *Positive* and *Romantic Intentions*.
- *Friendship MT:* if the characters are friends, they will desire to perform SEs with the *Positive Intention*.
- *Envy MT:* if "Farkas" is highly attracted to "Lydia" and believes that "Benor" is also attracted to "Lydia" or that "Lydia" is attracted by or admires "Benor", "Farkas" will desire to initiate *Negative Intention* SEs with "Benor". "Farkas" is envious of "Benor".
- *Memory Outcome MT:* the characters will avoid doing interactions that have previously ended with *Negative Outcome* and reluctant of doing the ones that ended with *Neutral Outcome*.
- *Memory Frequency MT:* characters will have a penalty on the volition of the last SEs that have been performed. Characters that have the trait *Obnoxious* have a smaller penalty.
- *Memory Decay MT:* if the SE has occurred a long time ago, this MT will lessen the Memory Frequency penalty.

3.1.5 Trigger Rules

Trigger Rules are rules that provoke secondary effects, applied after a SE has ended. After a character receives a new memory the Trigger Rules process the said memory (checking the participants, the SE that took place and the outcome) and modify the Social State. Akin to Microtheories, Trigger Rules illustrate social patterns and generalize that behaviour (e.g.: if “Lydia” hears “Farkas” insulting “Benor”, it is expected that they will like each other less and “Lydia” can decrease the *Friendship* Social Network Belief for those characters).

3.2 CiF Cycle

Now we will describe how the above mentioned components interact with each other when executing a CiF cycle. The stages of the cycle are the same as on the original CiF algorithm, adapted to the CiF-Ex components. All the structures and logic that coordinates the algorithm in our implementation will be defined in Chapter 4.

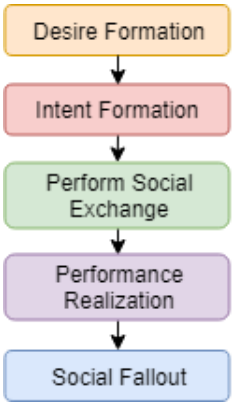


Figure 3.3: The CiF Cycle

3.2.1 Desire Formation

The first step of the algorithm decides **what SE will be executed and their participants**. Before this process starts the system checks if the number of on-going SEs is less than the defined maximum of concurrent SEs, if the CiF cycle is not on cooldown (after a SE has finished a timer is started that stops the cycle from staring, so the interactions don’t happen all in a row) and if the Player is in their local area (a fixed distance). If the maximum has not been reach, there’s no cooldown and the Player is nearby, the *Desire formation* process is started. When the process can’t be started, it aborts and is restarted after a cooldown.

For each character of the world we check if they are both not *Busy* and not *on cooldown*. If these conditions are validated, we can start the cycle to calculate the *initiator volitions*. The cycle iterates through each character and the Player, checking if they are busy; in case they are the cycle continues to the next character, if they aren't a new cycle that iterates through all Social Exchange space begins.

For each iteration of the cycle the initiator volition towards all available characters is calculated: first the *Pre-conditions* check if the SE can even be selected and output a temporary volition, then the *Microtheories* are checked and summed to the volition, lastly the *Initiator Influence Rules* are ran and their output added to the temporary volition, forming the initiator volition for that SE and receiver. We store the name of the SE and respective receiver that have the highest volition, else, we continue to the next iteration of the cycle.

In the end if no SE has positive volition, we do nothing and continue to the next character. Otherwise, we save the information of the highest volition SE and then continue to the next character. After all characters are checked, we verify which of the SE's has the greatest volition and select it to begin the next algorithm step. Both the initiator and receiver are marked as *Busy*. If no SE was chosen during this process, the Desire Formation step is started again after a timer ends. The pseudocode for the algorithm can be found in Algorithm A.1.

3.2.2 Intent Formation

In this step the Social Exchange structure is created. If the Player interacts with the initiator or the receiver before the initiator gets to the receiver the SE is canceled and the character that isn't interacting with the Player is unmarked as *Busy*. If the initiator manages to get close enough to the receiver the *Social Exchange* is created.

3.2.3 Perform Social Exchange

This step commences right after the previous step. The receiver volition is now calculated like in the *Desire Formation* step (the Pre-conditions aren't calculated for the receiver and the receiver volition are computed using the Receiver Influence Rules).

The *Pre-conditions* and *Microtheories* use conditions that verify *Social Network* values, *Social Network Belief* values, *SE memories*, *Relationships*, *traits* and *status*. The *Influence Rules* are based on conditions that examine *Relationships*, *traits* and *status*. After determined, the receiver volition value is compared to pre-defined thresholds, particular to the SE in question. This way the *Outcome* is asserted as *Positive*, *Neutral* or *Negative*.

3.2.4 Performance Realization

After the *Outcome* is decided the characters are ready to perform the appropriate *Instantiation*. Each *Social Exchange* has three animations, one for each possible outcome and different dialogue lines for each character and Player, depending on their role (initiator or receiver). When these are selected the *Instantiation* is instanced. Some *Social Exchanges* may also require additional steps (e.g.: if *Outcome* of the *GiveFood* SE is positive and the Player is one of the involved parties, it is required to transfer food from the initiator's inventory to the receiver's).

3.2.5 Social Fallout

The *Effects* related to the SE *Outcome* are applied to the *Social State* and character *Status*. A *memory* with the information of the SE is created, stored in the *Social Facts Knowledge Base* and broadcasted to all the characters in a certain range of the initiator (including the SE participators). The initiator and receiver are unmarked as *Busy* and the initiator is marked as being *On cooldown*. The CiF cycle cooldown begins.

Every character that has received a new memory now has to process them with the *Trigger Rules*, changing their *Social Network* values, *Social Network Beliefs* and possibly their *Relationships*.

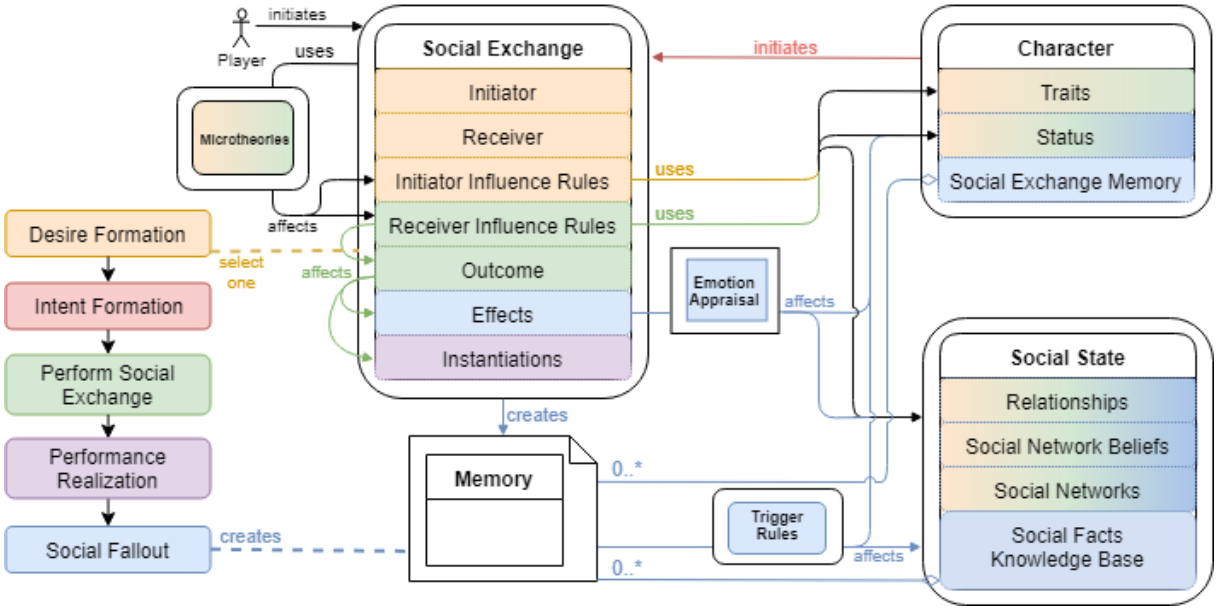


Figure 3.4: The CiF Cycle and CiF-Ex Components

4

Implementation

Contents

4.1 Social Exchanges 40
4.2 CiF NPCs 45
4.3 CiF GameState 50
4.4 Challenges and Difficulties 53

In this chapter we will discuss the implementation details of the CiF-Ex architecture in the video game *Conan Exiles* and some of the difficulties that were felt along the development process. The work was implemented using the Conan Exiles *DevKit*, a modification of the *Unreal Engine 4* that depends the engine's *Blueprints Visual Scripting* for coding, available for free on the *Epic Games Launcher*.

Most of the logic underlying the system is concentrated on three Blueprints: **BP_CIF_GameState**, **BP_CIF_NPC** and **CIF_SocialExchange_Component**. The Game State functions as the centralized controller, it runs CiF cycle, has references to all the CiF NPCs and delegates what Social Exchanges will start. The CiF NPCs store most of the social attributes and micromanage the NPCs (behaviour like the movement). The Social Exchanges handle the computational processes of the model.

4.1 Social Exchanges

Social Exchanges are structures that need to be initiated multiple times for each CiF cycle, as they are constructed and destroyed so frequently it makes sense to use a data type that is designed to add and remove behaviour on runtime, like the *UE4 actor components*. Social Exchanges are composed of:

- **Intention** is a keyword that gives semantic value to the SE (e.g.: the *Compliment* SE has the keyword *Positive*). The values can be *Positive*, *Negative*, *Romantic* and *Special* (used for the *DeclareLove* and *BreakUp* SEs). Intentions, represented by the enumeration *Enum_CIF_SE_Intention*, have default values for each SE is stored in the *DT_CIF_SE_Attributes* datatable (see Table B.6).
- **Initiator Volition** is a value used to prioritize what SE to execute.
- **Receiver Volition** is a value used to assert the receiver's response to the SE.
- **Pre-conditions** is a function implemented by only some SEs. This function tests if the Social Exchange should or should not happen (e.g.: the *Declare Love* SE should not happen if the NPCs are already dating) and aborts the SE in case it shouldn't.
- **Weights** represent the Influence Rules, floats that are multiplied by social parameters, generating volitions. There is one parameter for each *Social Network* variable (Goal, Current and Belief), *Trait*, *Status*, *Relationship* and an additional "Initial Value" parameter. There are two *Datatables*, *DT_CIF_Initiator_Weights* and *DT_CIF_Receiver_Weights* that have one row with these weights defined for each possible SE. Examples of the defined weights can be seen in Figure B.4.
- **Outcome** is an integer that represents the *Positive* Outcome when 1, the *Neutral* Outcome when 0, and the *Negative* Outcome when -1.

Social Exchanges have the following functions:

- **Microtheories** is ran after the Pre-conditions function. The implemented Microtheories are:
  - **Dating MT:** if the characters are dating, they will desire to perform SEs with the *Positive* and *Romantic Intentions*.
  - **Friendship MT:** if the characters are friends, they will desire to perform SEs with the *Positive Intention*.
  - **Envy MT:** the initiator will resent the receiver if the receiver and the character most liked by the initiator like each other. For this, the microtheory checks three conditions:
    * if the initiator believes that the receiver has *attraction value equal or greater than 7* for the other character;
    * if the initiator believes that the other character has *attraction value equal or greater than 7* for the receiver;
    * if the initiator believes that the other character has *admiration value equal or greater than 7* for the receiver.

    For each `True` condition a weight will be increased and used to reinforce the volition if the SE has a Negative Intention or to decrease the volition if the SE has a Positive Intention.
  - **Memory Outcome MT:** the character will filter their last interactions memories (five is used as default number), keeping the SEs that have the same name as the SE that they are now calculating and whose participators were themselves and their current SE partner. For each *Negative* Outcome the volition is decreased by five and for each *Neutral* Outcome the volition is decreased by three. The volition value is then returned.
  - **Memory Frequency MT:** the objective of this Microtheory is to stop the characters from repeating the same SE over and over again. For each SE in their memory that happened with their current SE partner the volition is decreased by 2 and if the SE in their memory is the same as the current SE the volition is decreased by 4. Characters with the *Trait Obnoxious* have a smaller penalty.
  - **Memory Decay MT:** this Microtheory lessens the penalty of the Memory Frequency MT based on time. For each SE in their memory that happened with their current SE partner if that SE has happened for longer than 180 seconds of in-game time the volition is increased by one and if it has happened for longer than 360 seconds of in-game time the volition is additionally increased by one.
- **Volition Function** is a weighted sum between a vector created using the Weights and the corresponding Social Network Values.

- **Assert Outcome** calculates the Outcome. The function compares the *Receiver Volition* to two thresholds specific to the current SE. If the volition is less than the *DownThreshold* the Outcome is *Negative*, if it is in-between the thresholds the Outcome is *Neutral* and if it greater than the *UpThreshold* the Outcome is *Positive*. The mentioned thresholds are stored in the *DT_CIF_SE_Attributes* datatable (see Table B.6).
- **Select Animation** returns the Conan Exiles animation that matches the current SE and the calculated Outcome. The animations are stored in the *DT_CIF_SE_Attributes* datatable (see Table B.6).
- **Execute Social Changes** performs the changes to the Social State and performs the *Emotion Appraisal* for the characters, according to the SE's Outcome. *Struct_CIF_Outcome_Weights* is a structure with float variables for each *Social Network* variables and *Status*. There are six datatables (three for the initiator and three for the receiver) that represent the three possible Outcomes (*DT_CIF_iPositive_Outcome*, *DT_CIF_rPositive_Outcome*, *DT_CIF_iNeutral_Outcome*, etc). The function selects the two Outcome datatables, obtains the rows corresponding the SE and adds the row entries to the Social State attributes.

The *Social Exchanges* that were implemented in this work are the following: *Introduce*, *Compliment*, *Insult*, *Brag*, *Flirt*, *SmallTalk*, *AskForFood*, *AskForDrink*, *OfferFood*, *OfferDrink*, *Date*, *DeclareLove* (if the Outcome is Positive, set the Dating Relationship), *BreakUp* (if the Outcome is Positive, unset the Dating Relationship), *NonLethalAttack* (the initiator physically attacks the receiver and the latter has the option to respond and attack the initiator back) and *ProtectOther*. A detailed view of the components can be seen in Table B.5. A short description of each SE can be seen in Table B.2.

The *Protect Other* SE happens when the initiator or the (NPC or Player) decides to warn the Receiver to stop *Insulting* or *Non-lethal attacking* another character. In the *Pre-conditions* function the initiator checks if they have any recent memories of the Receiver Insulting or Attacking other characters. If there are any cases, the Initiator assesses how much they like the abused characters, a process similar to the Volition Function calculation (there is a row for this propose in the weights' table), picks the most liked and adds the value to the SE's volition.

The *Social Exchange Attributes Structure* is a structure that is used to store SE default values, used during the SE process. An overview of this structure can be see in Table B.6.

4.1.1 CiF Dialogue Widget

The CiF Dialogue Widget (*W_CIF_NPCDialogue*) is the interface that the player uses to interact during Social Exchanges. The Table B.7 has brief view of this structure.

This widget has auxiliary functions that parse the authored dialogue lines.

- **Replace All *** function parses the dialogue lines always before they are displayed. The asterisk is used to encapsulate keywords that are replaced by character names:
  - ***npcname*** is replaced by the NPC Name variable.
  - ***playername*** is replaced by the Player's name.
  - ***othername*** is replaced by the Other NPC Name when the Social Exchange has a third participator (e.g: *Protect Other SE*).

  The **NPC Known Name** is the name that will be displayed to the Player on the Widget. If the *Introduce* Social Exchange has not happened between the NPC and the player, the NPC Known Name will be the nickname of the NPC. Nicknames are false names based on the NPC looks (e.g.: if a male NPC wears a helmet with horns the nickname could be "Horned Man") and are defined on the *DT_CIF_Nicknames* datatable.

- **Check Special Characters** function is used to parse the dialogue lines and find special characters that modify the Widget. The **#** special character indicates that the next three Strings on *Lines to Display* are player choices. The **$** special character indicates that the next *Dialogue Line* is one of the next three Strings on *Lines to Display*, depending on the *Outcome* of the current SE.

This interface has three sub-widgets:

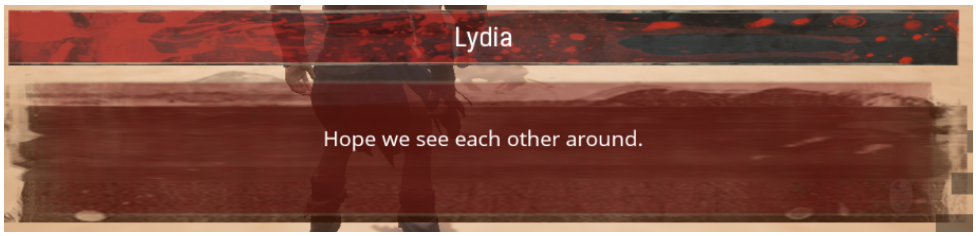
- **widget0:** *scroll box* with all the Social Exchanges that they can select;
- **widget1:** *text box* used to display dialogue lines from the NPCs;
- **widget2:** *vertical box* with three options that show three possible reactions to a SE.

A *Widget Switcher* (UE4 Widget element that allows to activate one of the widgets while the other remain inactive) is used to alternate between these three sub-widgets seen in Figure 4.1.

When the Player interacts with a NPC the *widget0* is initialized and displayed. Flavour text lines for all the SEs are stored in the *DT_CIF_SE_Player_Options* datatable (e.g.: the *Compliment* SE has "You are strong" and "You are honorable"). For each SE selection button it is select a random string from the above mentioned datatable.

Conditions are ran to check the SEs that the Player can perform: if the NPC and Player have already been introduced, the *Introduce* Social Exchange won't be available; if the Player and NPC are in a relationship the *Date* and *BreakUp* SEs are available, otherwise the *DeclareLove* SE appears; the Player inventory is verified, if there is at least one food item the *GiveFood* SE is accessible and if there is at least one drink item the *GiveDrink* SE is accessible. When one of these SEs is not possible, the button is still displayed but greyed out and unclickable.

(a) Widget0: Player dialogue selection



(b) Widget1: NPC's response

(c) Widget2: Player's response

Figure 4.1: Player to NPC interaction widgets

After the Player selects one SE, the widget is changed to *Widget1*. The SE is initialized and the Outcome is asserted. A block of dialogue lines is selected from the *DT_CIF_Dialogue_Responses* datatable, depending on the Outcome. This datatable has various possible responses with an arbitrary number of lines, for each different Outcome of each Social Exchange. The lines are then sequentially displayed in the *Widget1*. **The SE then ends and the Widget is closed.**

If the NPC starts the interaction with the Player, the *Widget1* is instantly displayed and lines are selected from the *DT_CIF_Dialogue_Initiations* datatable. This datatable has various possible initiations with an arbitrary number of lines for each Social Exchange. Lines are displayed until the *Check Special Characters* function detects a *#*, indicating that the Player now has to react to the SE. *The widget is now changed to Widget2* and three buttons corresponding to the three possible Outcomes displayed with dialogue lines. After the Player chooses, the Outcome of the SE is updated, the widget is changed back to *Widget1* and the lines reacting to the Player response are showed. The SE then ends and the Widget is closed.

4.2 CiF NPCs

A *Character*, component of the CiF architecture, is a structure that is associated with the NPC, as such, we took advantage of the already existing *BP_HumanoidDialogueNPC* and created a copy of it (that way we assured that this Blueprint would not be directly changed when the game updated, minimizing the risk of conflicts). This new Blueprint is called *BP_CIF_NPC* which serves as the base for the CiF *Character*, *Trigger Rules* and part of the *Social State* components, also controlling the logic for the NPCs. Table B.8 has an overview of this structure.

- **Social Attributes** is a structure that holds the social variables of the Character (like the *Traits* and *Status*) and the *Social Networks*. For more information on this structure see the Table B.9.
- **Social Exchange Memory** stores memories of SEs when the NPC is less than a defined distance to the SE initiator. One memory contains Strings for the names of the *Social Exchange*, *Initiator*, *Receiver*, *Origin* and *Other*; Integers for the *Outcome*, *SE ID* and *Timestamp*.
- **Social Network Beliefs** are matrices that represent the beliefs of Current Social Network values between different characters. The X and Y axes are Strings for the names of the characters. Each entry of the matrix is an Integer ranging from 0 to 10. If “Lydia” believes that “Benor” has *Friendship Social Network* value 3 for “Farkas”, the entry for the pair (Benor,Farkas) has the value 3.
- **Home Location** is a vector that represents the in-game world location of a CiF Bonfire, an object that functions as “home” for the CiF-Ex NPCs, restricting their movement to a radius around itself.

For more information about CiF Bonfires see Section 4.3.

- **Behaviour Tree** is a Behaviour Tree that controls the movement logic of the CiF NPCs. See Section 4.2.2 for more information.
- **CiF Dialogue Component** is used to call the CiF Dialogue Widget (see Section 4.1.1). The *CIF_Dialogue_Component* is based on the *Dialogue_Component* used by the Conan Exiles dialogue NPCs.

CiF NPCs have the following functions:

- **Go Rest** signals the NPC to walk up to the bonfire and begin resting.
- **Wander** compels the NPC to begin their default behaviour, wandering.
- **Walk to Object** makes the NPC to move itself to the specified game object.
- **Init Social Exchange** creates a new *Social Exchange Component* using the class of the SE that they are going to execute.

4.2.1 Initialization

The *Social Networks*, *Traits* and *Status* are initialized with the default parameters of the datatables, as explained in Section 4.2.3. If the *CiF Game State* has a *CiF Bonfire* reference the NPC will set their *Home Location* to the position of the bonfire, otherwise the variable will be set to the location where the NPC spawns. A weapon is put in the character's inventory.

The timer that controls NPC's CiF cycle is set with the periodicity bestowed by the character's traits (e.g.: *Shy*, NPCs will have a greater timer than others), when it ticks down the NPC can perform its first SE. The timer that decays the *Status* values is also activated. The *Physical Status* like *Hunger* or *Thirst* instead of being decreased periodically have a probability of being increased.

The *Tiredness* status is increased using another timer with a different period. This status is also increased by two every time a NPC performs a SE as the initiator. When the *Tiredness* status amounts to 10 (maximum value) the *Go Rest* event is activated. While resting the NPC can't start a SE with them as Initiator.

When the NPC is near enough to the bonfire, they perform a sitting animation, the *Tiredness* status begins to tick down and the NPC remains still until the status is 0 again. Then the *Wander* event is called. If the Game State does not have a bonfire reference, instead of moving to the bonfire to rest the NPC just sets the *Tiredness* status to 0.

4.2.2 General Behaviour

Each game tick the NPC checks if they have a new entry *Social Exchange Memory* and if they do, they process it. This process is equivalent to the *Trigger Rules* component of the CiF architecture. The NPC handles the new memory by checking the participators *names*, Social Exchange *name* and *Intention*. The *Social Network Beliefs* are changed during this process.

If the character was the *Other* character in a *ProtectOther* instance, in other words if they were being protected, their *Social Network* and *Status* will change, depending on the Outcome of the SE: for example, if the Outcome was *Negative* the *Emotion Status* of *Fear*, *Surprise* and *Sadness* will increase by two and the *Happiness* will decrease by two.

If the character participated in the SE and their *Social Network Current Friendship* value with another character rose to eight or above and the other character vice versa, the *Friends* relationship is set for them.

The *BT_CIF_Dialog* Behaviour Tree was a modification of the existing *BT_Dialog*. The logic inside the *BP_CIF_NPC* modifies the attributes used in the behaviour tree to select what nodes to execute. The *BT_CIF_Dialog* has 3 major tasks:

- **BTTaskWander:** the NPC finds a route in real time to a specific target.
- **CIF_SetNextWanderPoint:** a random location inside a radius from the bonfire is selected and set as the target to wander to.
- **BTTask_CIF_NewAttack:** the NPC attacks in front once with a basic attack. This is used in the *NonLethalAttack* Social Exchange.

When the NPC starts a SE as the initiator they will set the *BTTaskWander* target to the receiver and start to follow them. When they are within a defined distance, the initiator will perform a Conan Exiles ECharacter_Emote where they wave their hand at the receiver and the latter will stop and turn to the initiator. When a CiF NPC interacts with a character (either as the role of the initiator or the receiver) a message will appear on the top right of the screen, giving visual feedback related to the Social Exchange and its Outcome. The messages are defined in the *DT_CIF_SE_Attributes* datatable (see Section 4.1).

In the original Dialogue NPCs, when the Player pointed the mouse cursor to a NPC a Widget, *W_DialogueNPCInfo*, that showed the NPC's name and a instruction to interact with them appeared. This Widget was updated to the *W_CIF_DialogueNPCInfo*, seen in Figure 4.4. Instead of simply displaying the NPC name, the system checks if they have been introduced to the Player, displaying the NPC's nickname in the case they hadn't been. The *Emotion Status* of the NPC are checked and the highest valued one is chosen to appear. The NPC's *Social Emotion Status* regarding the Player are also

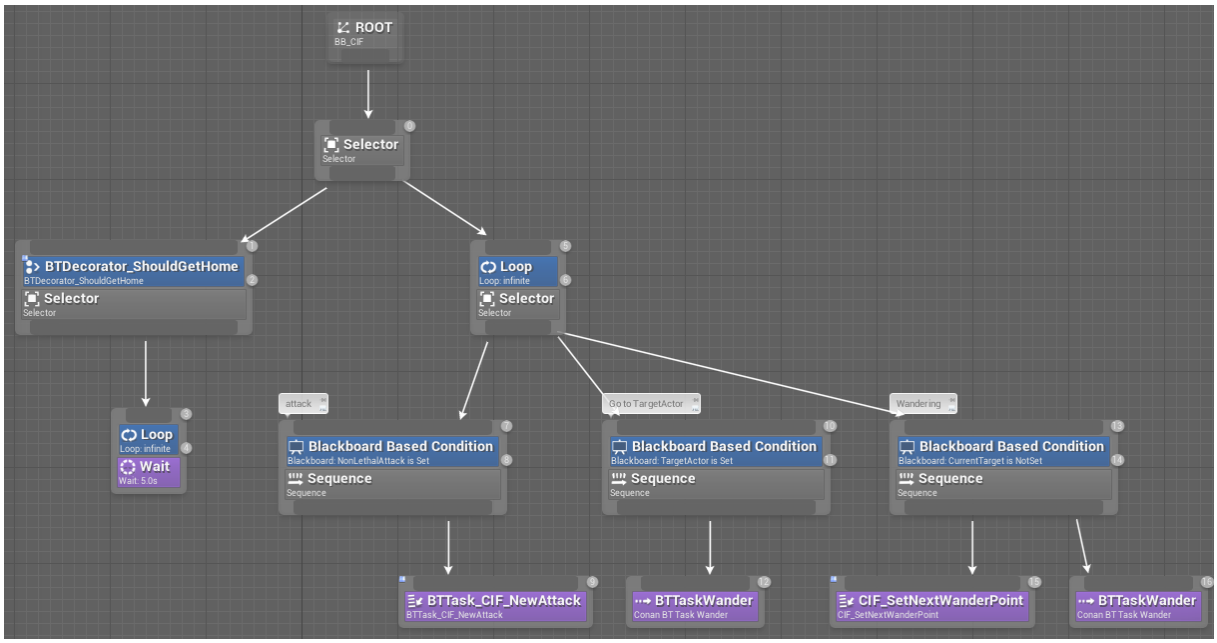


Figure 4.2: CiF Behaviour Tree

checked, the highest valued one that is not 0 is chosen and a sentence related to it appears (e.g.: if the NPC feels HappyFor the Player, the text “Feels happy for you” is displayed.

Emotion and Social Emotion Status gradually decay with the passing of time, the Physical status however increase as they represent physical needs of the character. The *Hunger* and *Thirst* status can be decreased by successfully performing SEs that grant them consumables (like the *Ask for Food SE*).

The *Tiredness* status can be decreased by *resting* by a bonfire. When resting the NPC won’t be able to initiate a SE, but other NPCs and the Player will be able to approach them and start a SE. Initiating SEs also add to a character’s *Tiredness*.

4.2.3 Social Attributes Structure

The **Social Attributes Structure** has most of the Character component variables used in the CiF-Ex architecture. This structure also includes the *Social Networks* (part of the Social State CiF-Ex component).

- **Social Networks**’ structure is divided in three variables: *Current*, *Goal* and *Belief*. Each variable is a map from String to Integer, in which the Key is a character name and the value is the actual value of the Social Network. The value ranges from -10 to 10. The datatables used to populate these Social Networks are matrices with the character names in both axes. Supposing that the pair (X,Y) in a datatable corresponds to the value V, we can say that: ”the character named X has V value for the character named Y” for that Social Network Variable.

Figure 4.3: Social Exchange interaction messages. On the top right it can be seen that the Player complimented Lydia and that the Outcome was Positive.

These datatables follow the naming convention of "DT_SN_SocialNetName_VarName" (e.g.: *DT_SN_Admiration_Belief*, *DT_SN_Friendship_Current*, etc). There are two sets of datatables for the Attraction Social Network, depending on the Player's gender.

- **Social Emotion Status** has maps from Strings (names of characters) to Integers (ranging from 0 to 10) for each social emotion status in the structure. The **Social Emotion Status** are: *Shame*, *Admiration*, *Reproach*, *Pity*, *Gloating*, *HappyFor* and *Resentment*. The Social Emotion Status are defined on the *DT_CIF_Social_Status* datatable. You can find an explanation of the Social Emotion Status in Table 3.1

  If "Lydia" does a SE that makes "Farkas" admire her, the value for the "Lydia" key in the *Admiration* map of "Farkas"'s Social Emotion Status variable will increase.

- **Emotion Status** and **Physical Status** have an Integer (ranging from 0 to 10) for each status in the respective structure. The **Emotion Status** are: *Surprise*, *Anger*, *Sadness*, *Happiness*, *Disgust* and *Fear*. The Emotion Status are defined on the *DT_CIF_Emotion_Status* datatable. See Table 3.2 for more information.

  The **Physical Status** are: *Hunger*, *Thirst* and *Tiredness*. The Physical Status are defined on the *DT_CIF_Physical_Status* datatable. See Table 3.3 for more information.

Figure 4.4: CiF Dialogue NPC Info Widget

If "Benor" witnesses a SE that makes them disgusted the value of the *Disgust* variable inside the Emotion Status will increase.

- **Traits** is a structure that stores all the traits in the form of Booleans. The traits are: *Friendly*, *Charming*, *Hostile*, *Shy*, *Gossiper*, *Obnoxious*, *Humble*, *Aggressive*, *Coward*, *Brave*, *Faithful*, *Unfaithful* and *Gender*. If "Farkas" is a coward, the *Coward* trait in their Traits variable will be `True`. Traits can be edited on the *DT_CIF_Traits* datatable. A short description of the Traits can be seen in Table B.4.

4.3 CiF GameState

The **BP_CIF_GameState** does the actual decision making for the CiF NPCs. It also contains the part of the *Social State* architecture component that the CiF NPC does not implement. An overview of this structure can be seen in Table B.10.

We have opted to centralize the NPC's decision making, similar to the CiF-CK implementation "GameManager". This way we can easily access all the characters and query their states, attributes, etc. A distributed approach would be preferable if we expected more complex behaviour like the NPCs physically looking for the other NPCs to check if they are busy, instead of automatically receiving that information from a centralized structure. As we did not aim for that kind of practice, the centralized option is easier to implement and debug.

- **Save Data** is used to serialize variables that would be lost if the game was turned off. Part of the *Social State* architecture component is implemented in this Blueprint. The Game State periodically

saves the Save Data. This is compromised of:

  - **Relationships:** a Map of String (NPC names) to *struct_CIF_Relationships*. This structure is a Map of String (NPC names) to *struct_CIF_Relationships_Types*, which contain Booleans to indicate if the NPCs are in relationships. *Relationships* is the representation for the Relationships Social State component of the CiF architecture.
  - **Memory Archive:** an Array of *struct_SE_Memory*. This Array stores all the Social Exchanges that have happened in the world. The *Memory Archive* is the representation for the *Social Facts Knowledge Base* Social State component of the CiF architecture.
  - **IDs:** a Map of String to Unique ID. The keys NPC names and the Unique IDs references to game objects (in this case, of the type *BP_CIF_NPC*), which can then be used to get the NPC itself.
  - **WorldTimer:** is an Integer. The value is the time in seconds since the world started.
  - **Current SE ID:** an Integer with the value of the latest SE ID. When a new SE begins this value is increased and used for that SE ID.

- **Next SE** (*Struct_CIF_Social_Exchange_Volition*) has information about the next SE to be performed. The structure stores the necessary SE data to initialize it: the SE, initiator and receiver names along with the initiator volition and SE ID.
- **Most Wanted SE** is a Map that associates a *Struct_CIF_Social_Exchange_Volition* structure to each character. This variable stores information about the SE with the highest initiator volition for every character.
- **Social Exchange Classes** is a Map whose keys are SE names and value is the corresponding class of Social Exchange Component. These classes are used to initiate the SE with the right class of component.
- **Ongoing SEs** keeps information about the SEs that are in process.
- **Bonfire** functions as a home to the CiF NPCs as they only move on a radius around the bonfire. When spawned, the bonfire automatically bounds itself to the GameState and broadcasts itself to the active NPCs, setting its physical location as the NPC's *Home Location* attribute. NPCs spawned after the bonfire are automatically bound to the bonfire.

The CiF GameState has the following events (Unreal Engine 4 special functions that run on their own thread):

- **Calculate Volitions** calculates the volitions for each character and populates the *Most Wanted SE* attribute.

- **Calculate Next SE** compares all the volitions for the SEs in *Most Wanted SE* and sets the one with the greatest volition to *Next SE*.
- **Start Most Wanted SE** starts the process for the *Next SE*.
- **Player Attack** used to force the Player to attack once in front of them. First the Player is armed, then a basic main hand attack is simulated and finally the Player weapon is unequipped again. This event is used in the *NonLethalAttack* SE.
- **Save Game** saves the CiF Save data to the file system.
- **Load Save** loads the CiF Save data from the file system or creates a new Save if there is none.

4.3.1 CiF Cycle

When the game starts the **CiF GameState** is initialized. First it loads a CiF Save from the file system, if none is found, a new Save is created. The starting relationships are loaded from the *DT_CIF_Relationships* datatable, that has the default relationships for the setting. The *Social Exchange classes* are populated. Lastly, the *World Counter Cycle* (a cycle that iterates each second the *World Timer* variable of the Save) is started along with the **CiF Cycle**.

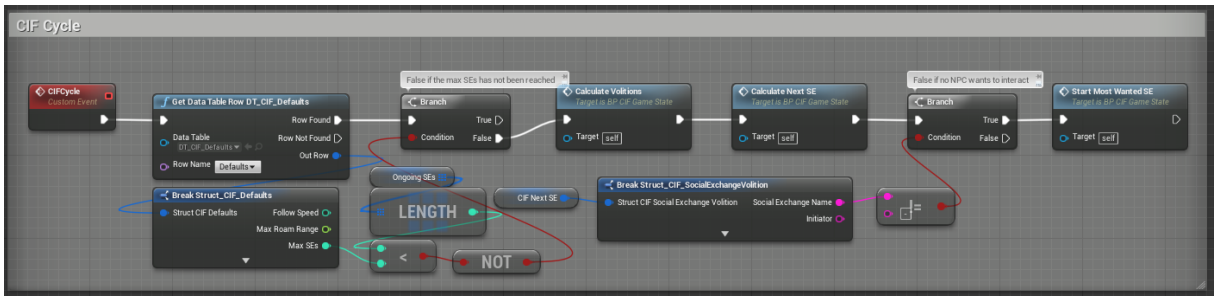


Figure 4.5: CiF Cycle Event

The CiF Cycle is ran periodically on a timer (the default is five seconds) and checks if the *maximum number of SEs* has been reached, in case it has, the event stops right away. In case the cycle can continue, the **Calculate Volitions** function is called.

In the *Calculate Volitions* function the Social Exchange with the greatest initiator volition for each character is calculated. The *Most Wanted SE* and *Next SE* attributes are set to their default values (so the system can detect if changes happen). Each character has each SE initiated with them as the initiator and the other living characters (plus the Player) as the receiver. The initiator volition is then calculated and if the new value is greater than the previous on *Most Wanted SE*, a new

struct_CIF_Social_Exchange_Volition structure is created with the information of the Social Exchange.

The **Calculate Next SE** function iterates through all the keys (the character names) in the *Most Wanted SE* Map. For each character it is checked whether their key value is different from the default, if the CiF NPC is ready to start a SE and not resting and if the character distance to the Player is not greater than a specified value (the default value is 1250).

Finally, the volition value of the new SE needs to be greater than 0 and greater than the volition in the *Next SE* variable. If these conditions apply, the *Next SE* is updated to the new one. After this function executes the cycle checks if *Next SE* has a new value in it and terminates otherwise.

The last step of the cycle is to run **Start Most Wanted SE**. The *Current SE ID* of the CiF Save is incremented, the *Next SE* is added to the *Ongoing SEs*. The *Init Social Exchange* event on the initiator CiF NPC is called with the *Next SE* information.

After the SE is started, the initiator needs to walk to the receiver to begin the performance of the SE. If the Player interacts with either the initiator or the receiver, the SE is aborted: the *GameState* calls the *Stop SE* event for each CiF NPC that participates in the SE, sets *Next SE* to the default value and removes the aborted SE from the *Ongoing SEs* Array.

4.4 Challenges and Difficulties

Conan Exiles was in development during all the implementation process of this work. This meant that from time to time the game was updated along with the DevKit. If we were to change anything in a Blueprint that was updated, our code could easily be overwritten and lost, so we had to be careful where changes were committed and keep the modification of Conan Exiles Blueprints to a minimum, creating a new one was always preferable.

Unfortunately, even being careful about this aspect was not enough to avoid setbacks, occasionally after an update Blueprints that we had never modified would stop compiling and parts of them would be missing. The two ways to solve this problem were to either try to debug and fix the error or try the Epic Games Launcher “Validate Files” option (that could take more than two hours to run). Blueprints in the higher hierarchy of the ones that we create could also be updated and cause trouble down the line.

During the development, three Blueprints got corrupted. It is documented by the UE4 community that the Structures are unstable and could lead to crashes and even Blueprint corruption (we are unsure if this is aggravated because the DevKit runs an older version of the Unreal Engine). Structures would get buggy when modified after they had been already used in other Blueprints. When a Blueprint is

corrupted, whenever it is opened or another Blueprint that references it is opened, the editor crashes. The solution that the community recommends to this problem is to open individual Blueprints and start filtering the ones that don't cause the editor to crash. The DevKit is a heavy piece of software, so anytime that it crashed and had to be restarted the process could take several minutes. After the problematic Blueprint was found, the solution would be to either replace it with one backup (sometimes even backups would be corrupted) or to re-implement it.

One limitation of the DevKit is that it denies access to the video game's C++ code and to new code creation, so many functions were a *blackbox* to us and what actually happened inside them had to be guessed. When unexpected behaviour originated inside one closed Blueprint it could take entire days of work to find out what was really happening.

Datatables were also a problem to edit, despite displaying a matrix, the cells can't be directly altered, you have to edit them on a vertical scroll list that has drop down menus if the data type has more than one layer (e.g.: a Structure that has another Structure inside, see Figure 4.6). This is not practical to visualize nor edit data in bulks (like our volition or dialogue datatables). The option to export as a CSV or JSON file was also not practical because when using Structures the Unreal Engine adds a generated label for the Structure, making it hard to read and modify, as seen in Figure 4.7.

Figure 4.6: The UE4 interface for editing datatables

Figure 4.7: Exported datatable to CSV format

As such we developed a Python script that parses Excel files (easy to edit) and exports them to the CSV format, able to be imported by the DevKit. Figure 4.8 shows how the same lines in Figure 4.6 can be edited in an Excel file.

	Introduce	Compliment	Insult	Brag
Default_Positive	Pleasure to meet you.	Thank you!	Why would you say that?	So impressive!
	Hope we see each other around.	You are very kind.	I am sad.	I knew you had it in you.
Default_Neutral	Hi.	Ok.	I don't get you.	Oh, ok.
	Hey.	Sure.	Sure.	Sure, cool.
Default_Negative	You've made my day worse.	Get away from me.	Go eat a club.	What a narcissist.
	player name? More like, Ugly Face.	Ewh.	No you.	Go dream elsewhere.
	Pleasure to meet you.			

Figure 4.8: Excel file for the Dialogue Responses

5

Evaluation

Contents

5.1 Scenario 58
5.2 Procedure 61
5.3 Metrics 62
5.4 Results 62

Our hypothesis are that, our social interaction model will:

- improve Conan Exile's **User Experience**;
- improve Conan Exile's **NPC Believability**.

A controlled experience was conducted to validate our objectives. For this we performed a **repeated measures** experiment where the user played the same scenario with **two different conditions**: the **Experimental Condition**, where the participant was able to interact with three CiF-Ex NPCs; and the **Control Condition**, where the participant played Conan Exiles and interacted with three NPCs, as they are in the video game.

Each user was requested to fill questionnaires that allowed us to collect data needed to validate our objectives.

5.1 Scenario

To limit the user's focus to a specific area, they were restricted to explore one section of the Conan Exiles' map, a beginner's region near the unmodded game's first spawn. The area was large enough that the user experience of playing Conan Exiles was still preserved.

The user assumed the role of an exile named "Nemon", in possession of some basic equipment. When they began their journey a camp with three NPCs could be seen on the flat land in front of them. The user was equipped right from the beginning so they could skip the initial step of grinding basic tools and consumable resources.

The Player would talk with at least one of the NPCs and then had the choice to continue the interaction if they wanted to, influenced only by the appeal that the initial interaction provoked.

5.1.1 Control Condition - Conan Exiles Dialogue NPCs

This condition used the Conan Exiles original Dialogue NPCs. The used NPCs were:

- **Muriela the Artisan:** the priestess of Mitra. Muriela says that she has been chosen by her god, Mitra, to create a giant statue of himself. She says that Mitra is a gentle and kind god that protects wisdom and knowledge.
- **Arcos the Wanderer:** an experienced exile that warns the Player about the perils and mysteries of the wasteland.
- **Mek-Kamoses:** tells the Player of his past and his ambition: to restore his sorcery powers with the ring of the god Seth. After being interacted with four times he promises to teach the Player sorcery and to raise an army with them.

Figure 5.1: User study spawn. The NPC camp can be seen close to the Player.

The characters wandered around the bonfire up to a defined distance and stopped when the Player interacted with them. These NPCs had voice acting complementing their dialogue lines and shared bits of lore of the world. When the Player *talked* with them, each character had 5 different interactions (with the exception of Arcos that only had 4) that could last up to around 30 seconds.

5.1.2 Experimental Condition - CiF-Ex NPCs

For this condition three CiF-Ex NPCs were created: *Lydia*, *Benor* and *Farkas*. The physical aspect of the characters was the same of their counterparts in the *Control Condition*. As CiF-CK was one work important to the conceptualization of our system, we payed an homage and named our characters with the same name as three Skyrim NPCs. Each of these CiF-Ex NPCs supported 15 Social Exchanges, had different dialogue lines for each SE (some also inspired by the Skyrim NPCs), 12 traits, 2 possible relationships, 16 Status and 3 social networks. The status, social network beliefs and SE Memory start with their default values.

The NPCs wandered around one bonfire, where they could sit nearby and rest. The CiF-Ex were capable of initiating Social Exchanges (small dialogues) with other NPCs or with the Player, to which they would then respond. The Player was also able to start the same Social Exchanges and watch the NPCs react to them, depending on past interactions and the NPC's personality.

Benor is *humble*, *brave* and *faithful*, making him someone that treasures his relationships. *Lydia* has the traits of *friendly*, *gossiper*, *obnoxious* and *brave* that when combined form someone that won't

	Traits												
	Friendly	Charming	Hostile	Shy	Gossiper	Obnoxious	Humble	Smug	Aggressive	Coward	Brave	Faithful	Unfaithful
Benor							✓				✓	✓	
Lydia	✓				✓	✓					✓	✓	
Farkas			✓					✓	✓				✓

Figure 5.2: Study's Trait values

hesitate to start a SE (or to repeat it). *Farkas* is *hostile*, *smug*, *aggressive* and *unfaithful*, a personality that makes him prone to start SEs with *negative* intention.

	Friendship											
	Current				Goal				Belief			
	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas
Player		0	0	0		0	0	0		0	0	0
Benor	0		8	8	0		6	7	0		6	7
Lydia	0	5		2	0	5		2	0	5		4
Farkas	0	8	4		0	6	3		0	6	5	

Figure 5.3: Study's Friendship Social Network values

As can be seen in the Figure 5.3, *Benor* feels that he is good friends with everyone and that the others also like him fairly. His goal is to slightly decrease his Friendship with them. *Lydia* thinks reasonably of *Benor* and more or less of *Farkas* and does not want want that to change (the current value is the same as the goal). *Farkas* thinks fairly of the others but wants his friendship with Lydia to decrease. *Benor* and *Farkas* have the relationship *Friendship* set to `True` because both of their current Friendship values for the other is greater or equal to 8.

	Admiration											
	Current				Goal				Belief			
	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas
Player		0	0	0		0	0	0		0	0	0
Benor	0		6	3	0		6	3	0		5	3
Lydia	0	5		3	0	6		2	0	5		5
Farkas	0	2	4		0	2	5		0	2	5	

(a) Study's Admiration Social Network values

	Attraction											
	Current				Goal				Belief			
	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas	Player	Benor	Lydia	Farkas
Player		-6	0	-6		-6	0	-6		-6	0	-6
Benor	-6		7	-6	-5		8	-6	-6		4	-6
Lydia	4	5		3	4	6		4	6	6		6
Farkas	-6	-6	6		-6	-6	7		-6	-6	6	

(b) Study's Attraction Social Network values

Figure 5.4: Study's Admiration and Attraction Social Network values

The tables for the *Admiration* and *Attraction* Social Networks are in the Figure 5.4. The values for the Player are 0 because they don't know him, except for the Attraction Social Network, to simulate that there is an initial attraction purely based on the looks of the characters.

It is relatively simple to create a social panorama with backstories influenced by past social interactions that organically spur different storylines for each Player. It is up to the author to create a setting with as much detail as they desire.

5.2 Procedure

Ahead of starting the experiment, the user was asked to fill a questionnaire that traced their basic profile. Then they would be briefed on the survival video game genre (concepts like hunger, thirst, stamina, crafting, etc) and Conan Exiles (a summarized explanation of the setting and the basic controls, which were also clearly displayed in the video game interface) in case they had no knowledge of either or both.

Before seizing the controls of their character the user was told that their only objective was to talk to at least one of the exiles in front of them and to recall their name afterwards. They were told that they could interact with the NPCs whenever they wanted during their play time and that they were free to explore the map and do anything they pleased. The objective of memorizing the NPC's name was merely to give the user an objective that assured that the Player interacted with at least one NPC.

Afterwards a 15 minutes play session of one of the conditions would start. The user was not told how long they would be playing for, simply that we would notify them to stop. The initial condition was alternated, so 50% of the users played the *Experimental Condition* as the first condition and the other 50% played the *Control Condition*.

When the 15 minutes had passed, the user would fill a questionnaire related to their testing. Without informing the user, the condition would be switched and then they would be asked to play the game once again for awhile. When the time was up, they would be asked to fill the final questionnaire (concerning exclusively their last session). This was identical to the first but with an additional question to detail which was the session that they had enjoyed interacting with the NPCs the most (and also an optional open question to grant some feedback and commentaries). After the user was done the ones that manifested interest in knowing more about the study or that wanted to provide some direct feedback were gladly heard.

5.3 Metrics

After each playthrough the user completed a survey that collects the necessary information about metrics to test our objectives. Besides these two questionnaires, an in-game metric was collected.

5.3.1 User Experience

The *User Experience* was measured using four different metrics. The **Enjoyment** illustrates how much the user likes playing the game and was assessed using the *Game User Experience Satisfaction Scale - GUESS* [23]. The **Flow** depicts how engaged the users were in the game and was measured using the *Flow Short Scale* questionnaire [24]. The survey had one question that did not belong to the above mentioned dimensions but was also used to help analyzing the *User Experience*: "("In what session did you enjoy interacting with the NPCs the most?). We refer to this as the **Condition Preference**. Besides these questionnaires, an in-game metric was collected while the user played, the **Bonfire Time**. Whenever the Player character was within a radius from the bonfire a variable would increment each second. We used this metric to test if there was a correlation between the used condition and the time spent in the proximity of the bonfire.

If these metrics improve when the users test our condition we will have a good indication that the **User Experience** has also improved.

5.3.2 NPC Believability

The **Believability** represents how believable the NPCs' behaviour is. This metric was measured using an adaptation of the *Believability questionnaire* [25].

One interesting detail to note is that the question number 9, "The NPCs' behaviour was predictable", was analyzed independently from the rest of the questions (but was still used to evaluate the *Believability*). This metric is referred as the **Predictability**. Unlike the other questions, obtaining lower results for this question is not necessarily a bad outcome, since a behaviour too predictable is not fun to interact with: a degree of uncertainty provides a more believable interaction. However the value for this metric should not be minimized as it would mean that the NPCs' behaviour was erratic. We believe that a value slightly greater than the middle of the scale should be optimal.

5.4 Results

In this section we will process the collected data and interpret it. The result of this interpretation will allow us to verify if the variation of both conditions created statistically significant differences that validate our hypotheses.

5.4.1 Questionnaire answers

Here we will present an overview of the answers given by the users.

5.4.1.A Profile Questionnaire

This questionnaire allowed us to have an idea of the characteristics of our population of 25 participants. Of these participants, 92% were male and 8% were female. Regarding the age, most were in the ages of 22 to 25 years old. Only one participant was not a gamer, most said that they play an average of 6 hours to 12 hours a week or that played for about 13 hours to 20 hours. A major 92% of the participants said that they were familiar with the "survival" genre before. 88% of the users answered that they had played a survival video game before, only 16% however answered that they had played Conan Exiles before. For 48% of the users the *Experimental Condition* was the first tested condition.

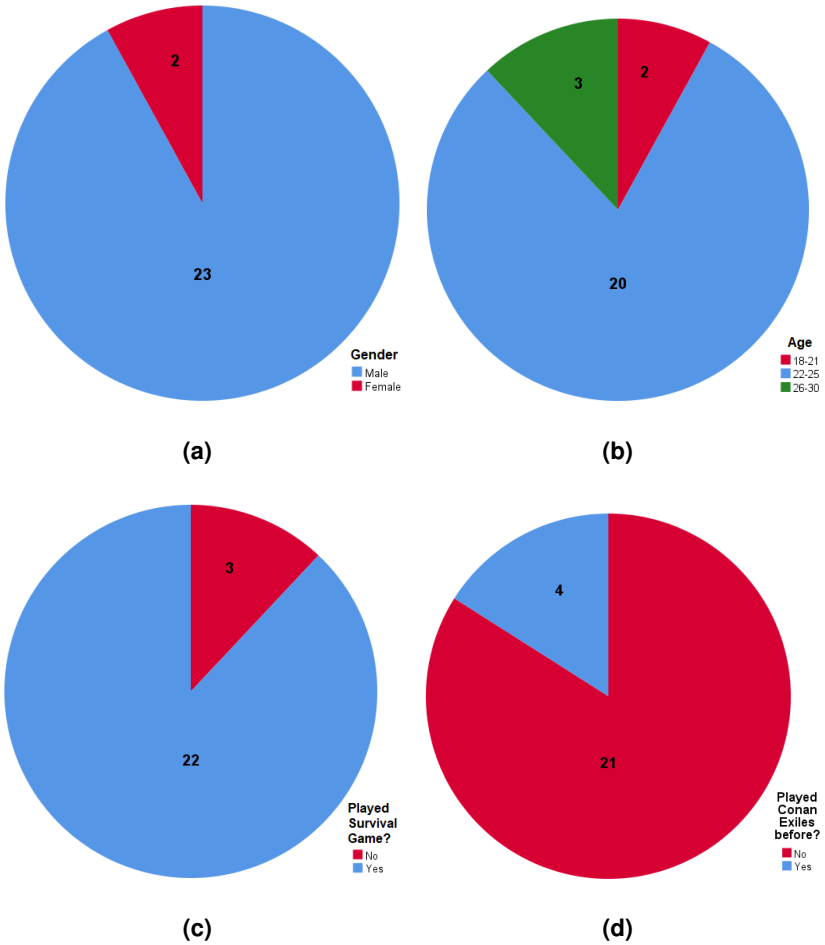


Figure 5.5: Profile questionnaire pie charts: (a) gender; (b) age; (c) played a survival game before; and, (d) played Conan Exiles before.

5.4.1.B Metrics Questionnaire

The *Metrics questionnaire* was separated in three parts, each intended to collect data on the different metrics, *Enjoyment*, *Flow* and *Believability*. The participant answered this questionnaire twice, each at the end of a playing session. In the end of the second Metrics questionnaire, the participant was also asked in which playing session they enjoyed talking to the NPCs the most. The metrics were measured using a Likert Scale from 1 to 7. We asked the users to point how much they agreed with each sentence. The direct results to the questions for the *Enjoyment*, *Flow* and *Believability* can be seen in Figure B.3, Figure B.1 and Figure B.2, respectively.

5.4.2 Data Analysis

The first step was to conduct *Cronbach's Alpha* test for the Enjoyment, Flow and Believability to see if there was acceptable inter-relatedness to aggregate each dimension on a single *mean* variable. The Cronbach's Alpha values for each metric and condition can be consulted in Figure 5.6.

		Condition	
		A	B
Metric	Enjoyment	0.871	0.866
	Flow	0.845	0.779
	Believability	0.907	0.867

Figure 5.6: Cronbach's Alpha results for all the metrics and conditions

As all the values were greater or equal to the acceptable threshold of 0.700 (and most greater than 0.800), **all the dimensions were aggregated**. The second question of the Enjoyment questionnaire, "I felt bored while playing the game", had a significantly lower corrected item-total correlation value for both conditions. This could have been caused because the question was the only with inverted measures and some users may have missed that detail.

Next we applied *Normality tests* to the aggregated means to verify if the data was parametric or non-parametric. The *p-values* for **Enjoyment Mean, Believability Mean and Bonfire Time Mean were greater than 0.05, so the test to be conducted would be parametric**. The parametric test used was the **Paired Sample t-test**.

The p-values for the *Experimental Condition*'s Flow mean and the *Control Condition*'s Predictability mean were less than 0.05, so **the tests would be non-parametric for the Flow and the Predictability means**. The non-parametric test was the **Wilcoxon signed-rank test**. For the last question of the survey, where the users explicitly chose their favourite condition, a **Binomial** test using a test proportion of 50% was conducted.

For the successful tests we have also calculated the **effect size**, using the boundaries: [0, 0.2[is a weak effect, [0.2, 0.5[is a moderate effect and +0.5 is strong effect.

5.4.2.A User Experience

Our hypothesis is that the *User Experience* will improve with the *Experimental Condition*. The four metrics used to verify this assumption were the *Enjoyment*, *Flow*, *Bonfire Time* and *Condition Preference*.

For the Enjoyment chart, Figure 5.7, the first impression is that the *Experimental Condition* seems to have gotten slightly better results in all the questions. Note that the question "I felt bored playing the game" had the measures inverted, which could explain why this question had better results for the *Control Condition*, despite the other questions seemingly pointing to a more enjoyable *Experimental Condition*. The Flow chart, Figure 5.8, appears to not show any clear indications of a trend between the versions.

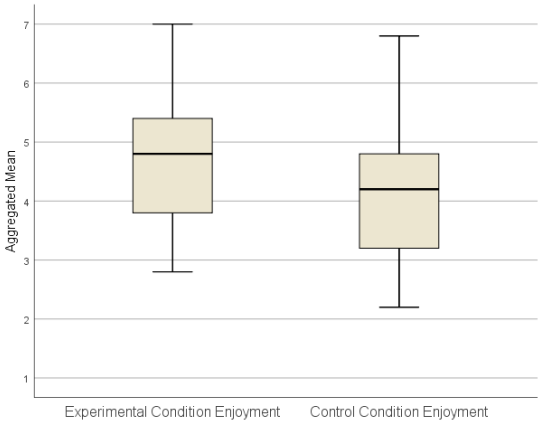


Figure 5.7: Mean bar chart for the *Enjoyment*

The **Enjoyment** *(p-value = 0.007; r = 0.512)*, **Bonfire Time** *(p-value = 0.001; r = 0.623)* and **Condition Preference** *(p-value = 0.043)* metrics **succeeded in showing positive statistical significance** as their *p-value* is less than 0.05. The **Flow** *(p-value = 0.584)* metric **did not show statistical significance**.

The *Enjoyment* and *Bonfire Time* metrics exhibit strong effect sizes. The *Bonfire Time* mean improved from 267 seconds to 419 seconds, an increase of 57%, Figure 5.9. 72% of the participants preferred the *Experimental Condition* over the *Control Condition*, Figure 5.10. We can conclude that **these metrics improved, hinting that the User Experience has also improved** for the *Experimental Condition*.

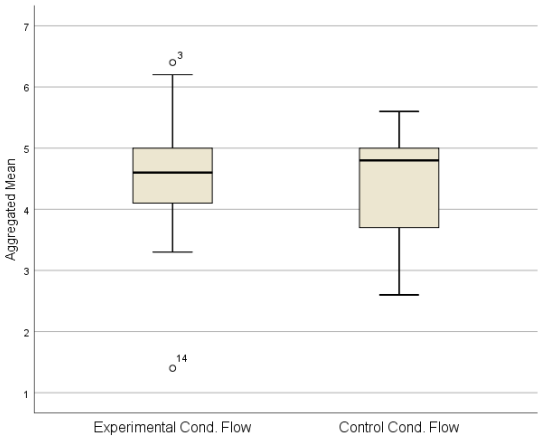


Figure 5.8: Mean bar chart for the *Flow*

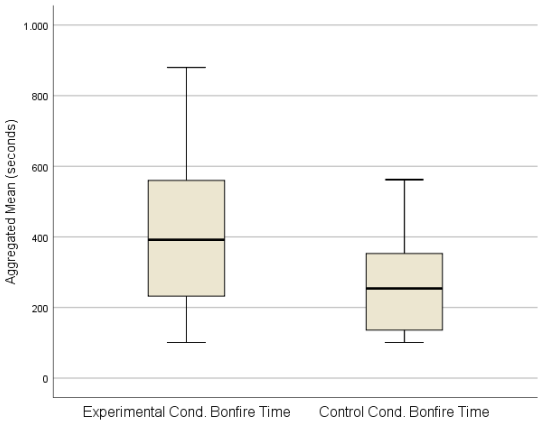


Figure 5.9: Mean bar chart for the *Bonfire Time*

5.4.2.B NPC Believability

Our hypothesis is that the *NPC Believability* improved for our condition. At first glance the Believability chart, Figure 5.11 for the *Experimental Condition* appears to reveal slight improvements.

The **Believability** *(p-value = 0.013; r = 0.482)* metric registered **positive statistical significance**. The effect size of the *Believability* metric is **moderate**, but very close to the strong effect size threshold of +0.5 for the strong effect size. **The obtained results demonstrate that the *NPC Believability* improved** when the users tested the *Experimental Condition*.

The **Predictability** *(p-value = 0.012; r = 0.5018)* was *statistically significant* and had a **strong effect size**. The value was lower and nearer to the middle value of the scale for the *Experimental Condition*. This result suggests that the *Experimental Condition* characters were less predictable.

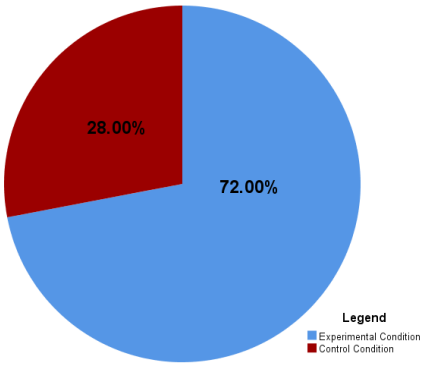


Figure 5.10: Pie chart for the *Condition Preference*

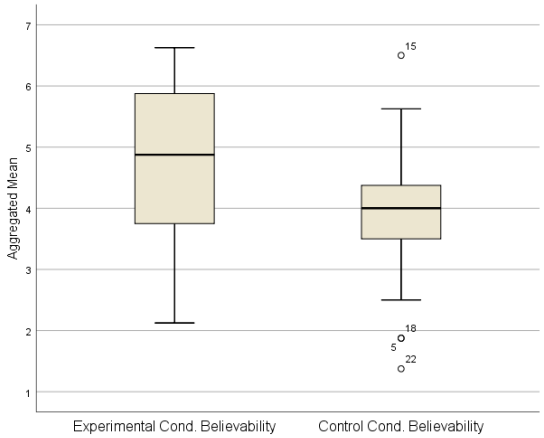


Figure 5.11: Mean bar chart for the *Believability*

5.4.3 Discussion

The obtained results allow us to say that our objectives were reached. The *Enjoyment*, *Bonfire Time* and *Condition Preference* metrics displayed positive statistical significance for the *Experimental Condition*, despite the quality voice acting and dialogue lines that the *Control Condition*'s NPCs had. This result is an **indication that the User Experience has also improved**. This conclusion is corroborated by the percentage of **users that preferred the CiF-Ex NPCs to the original NPCs, 72%**.

The ***Bonfire Time* mean was increased from 267 seconds to 419 seconds, an increase of 57%**. The inflation of this metric on our condition implies that the users spent more time on the area neighboring that homes the CiF-Ex NPCs, entailing that, after implementing CiF-Ex, adding these characters to the scenario could be a relatively easy way to raise the gameplay value of a location without changing much of the layout of the environment. The improvement for this metric hints that the users were interacting with the CiF-Ex NPCs for longer periods of time, despite that one interaction with a Conan Exiles Dialogue NPC could last for a bit longer than 30 seconds and the implemented interactions with the CiF-Ex NPCs last for a small portion of that time.

Despite this, the *Flow* dimension did not reveal a significant statistical difference. Even though this was not the best case scenario, this also means that at least the *Flow* did not deteriorate, so our condition did not disrupt the Conan Exiles experience. This could be explained by the narrow amount of time the users played the game for. Most of the participants (84%) were playing Conan Exiles for the first time, so it's expected that by playing it twice for 15 minutes they would still be in a phase where every aspect of the game is fresh and interesting, and that they wouldn't feel bored. As such, the full effects of our system wouldn't be evident in these test conditions. We believe that if the users tried the video game for longer time periods the *Flow* metric would also be higher for our condition.

The collected data shows us that the CiF-Ex NPCs managed to be seen as more **believable and less predictable**, which may lead to a more engaging, diverse experience. We can affirm that the **NPC Believability was enhanced**.

6

Conclusion

Contents

6.1 Future Work 71

As Artificial Intelligence is becoming common place in our lives and video games aim to release meaningful and realistic experiences, we set the challenge of **improving the NPC Believability** as well as **improving the User Experience** for the AAA video game Conan Exiles by **implementing a Social AI Architecture**, providing autonomous agents with their own personality, beliefs, desires and relationships. This new model would be referred as **Comme il Faut - Exiles (CiF-Ex)**.

By design, the *Comme il Faut* architecture aims to grant NPCs with the social behaviour that we required, keeping the authoring stress as low as possible. With this in mind we set on creating a new member of this family, greatly inspired by the CiF-CK architecture.

The tool used to implement our system, the Conan Exiles DevKit, a highly complex system that provides the means to modify the base video game. Despite the challenge of working with DevKit, it undoubtedly allowed us to make an implementation of our solution that features:

- **Characters** with their own personality traits, social networks and status, making choices according not only to their state but also to the world's past and present context, the social state;
- **Emotions**, based on the OCC model of emotions, that fluctuate with the Social Exchange's outcome and deeply influence the character's intent;
- **Social Network Belief system** that models the Social Network values that the characters assume, not only from other NPCs to the owner of the belief, but also what one NPC believes that two two NPCs feel for each other;
- **Social Exchange outcomes that are not binary**, each SE can have a *Positive*, a *Neutral* or a *Negative* conclusion to the initiator instead of only two proposed in other CiF architectures;
- **Freedom** for the Player to start any Social Exchange and choose how to react to the NPCs' interactions;
- **Social Exchanges** that are not tied to any specific character, so they can be easily generalized, authored and targeted to any character of the world;
- **Dynamic social world** with its own norms and background that changes with the actions of the different characters.

The **User Experience** was measured with the *Enjoyment*, *Flow*, *Bonfire Time* and *Condition Preference* metrics. All the metrics were improved in our condition, however only the ***Enjoyment*, *Bonfire Time* and *Condition Preference* showed statistical significance**. The ***Bonfire Time* mean increased from 267 seconds to 419 seconds, an improvement of 57%, while 72% of the study participants preferred the CiF-Ex characters** to the original ones.

The **NPC Believability** obtained greater results and positive statistical significance for the *Experimental Condition*, as such, we can suggest that the **CiF-Ex NPCs were more *believable***.

We are convict that we have demonstrated the success of the CiF-Ex model. The results highly suggest that all of the objectives that we had set to accomplish were achieved: the **CiF-Ex NPCs were perceived as more believable and improved the user experience**. The NPCs were also less predictable than the unmodified NPCs.

CiF-Ex is an extension of the CiF model and its implementation is not restricted to Conan Exiles. We believe that our model would obtain favourable results in other environments, be it a video game or a different kind of application (AI assistants, companion robots, etc). A paper about this work has been accepted to be present in the conference Foundations of Digital Games 2019, San Luis Obispo, California, USA.

6.1 Future Work

The obvious way to improve the work that has already been done would be to author more characters, traits, social exchanges, status, microtheories and trigger rules and improve the existing. *Voice acting* is one missing feature that some testers manifested interest in. We expect that this would improve the *User Experience* and the *Believability* of the NPCs.

Social Exchange memory sharing between the characters (like gossiping, telling a happy memory, etc) would extend the memory capabilities of the NPCs. Alea [26] provides a conceptual model of autobiographical memory sharing for social purposes. The mentioned work could serve as base to design this feature.

With memory sharing added to the model *lying* would also be an interesting concept to explore. NPCs would lie to try to manipulate other NPCs' opinions, backed by the capabilities of the Social Network Belief system. For example, using the already existing *Envy Microtheory*, NPCs could target characters that they envied and slander them to their most loved ones, undermining their social position.

The addition of the *Gifting social exchange* could be complemented by the implementation of the *Cultural Knowledge Base*, present in the CiF architecture. The *Cultural Knowledge Base* would give cultural connotations to the world objects, adding new dialogue possibilities, as characters could discuss their likes and dislikes.

The bonfire system gave origin to an interesting idea: various CiF camps with social NPCs could

be created, each with their bonfire that marks their limits. If a NPC was being harassed or did not like the characters residing in their current bonfire, they could try their luck and migrate to another camp. It would be interesting to observe what kind of social dynamics this would cause. Would NPCs of one personality type converge in the same camps? Would the migration never stop? It could also be interesting to analyze the human factor, would the Player harass and cast out characters that they did not like? Would they create a new camp that attracts NPCs?

Bibliography

[1] Funcom, “Conan Exiles,” Microsoft Windows, Playstation 4, Xbox One, 2018.

[2] J. McCoy, M. Treanor, B. Samuel, N. Wardrip-Fruin, and M. Mateas, “Comme il Faut: A System for Authoring Playable Social Models.” in *AIIDE*, V. Bulitko and M. O. Riedl, Eds. The AAAI Press, 2011. [Online]. Available: http://dblp.uni-trier.de/db/conf/aiide/aiide2011.html#McCoyTSWM11

[3] M. Guimaraes, P. Santos, and A. Jhala, “Cif-ck: An architecture for social npcs in commercial games,” in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2017, pp. 126–133.

[4] J. McCoy, M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin, “Social Story Worlds With Comme il Faut,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 2, pp. 97–112, June 2014.

[5] M. C. G. M. Guimarães, “Skyrim Mod for Social NPCs,” Master’s thesis, Instituto Superior Técnico, 2016.

[6] J. Dias and A. Paiva, “Feeling and Reasoning: a Computational Model for Emotional Agents,” in *Proceedings of 12th Portuguese Conference on Artificial Intelligence, EPIA 2005*. Springer, 2005, pp. 127–140.

[7] D. V. Pynadath, “PsychSim: Agent-based Modeling of Social Interactions and Influence.” *Proceedings of the International Conference on Cognitive Modelling, ICCM 2004*, pp. 243–248, 2004.

[8] R. Prada and A. Paiva, *Social Intelligence in Virtual Groups*, nadia magnenat-thalmann, lakhmi c. jain, nikhil ichalkaranje ed., ser. Studies in Computational Intelligence. Springer Berlin / Heidelberg, September 2008, vol. 140, ch. New Advances in Virtual Humans: Artificial Intelligence Environment, pp. 113–132.

[9] M. O. Ferreira, “A Merchant Model,” Master’s thesis, Instituto Superior Técnico, 2017.

[10] A. S. F. Machado, “A procedural quest generator for Conan Exiles,” Master’s thesis, Instituto Superior Técnico, 2017.

[11] CD Projekt Red, “The Witcher 3: Wild Hunt,” Microsoft Windows, Playstation 4, Xbox One, 2015.

[12] Bethesda Game Studios, “The Elder Scrolls V: Skyrim,” Microsoft Windows, Nintendo Switch, Playstation 3, Playstation 4, Xbox 360, Xbox One, 2011.

[13] R. E. Howard, *Conan the Conqueror*. Gnome Press, 1950.

[14] M. Treanor, J. McCoy, and A. Sullivan, “A framework for playable social dialogue,” 2016. [Online]. Available: https://aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14018

[15] Bethesda Game Studios, “Fallout 4,” Microsoft Windows, Playstation 4, Xbox One, 2015.

[16] A. Ortony, G. L. Clore, and A. Collins, *The Cognitive Structure of Emotions*. Cambridge University Press, 1988.

[17] J. Dias, S. Mascarenhas, and A. Paiva, “FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework,” in *Workshop on Standards in Emotion Modeling*, Leiden, 2011.

[18] A. Popescu, J. Broekens, and M. van Someren, “Gamygdala: An emotion engine for games,” *IEEE Transactions on Affective Computing*, vol. 5, no. 1, pp. 32–44, Jan 2014.

[19] P. Fletcher, F. Happé, U. Frith, S. Baker, R. Dolan, R. Frackowiak, and C. Frith, “Other minds in the brain: a functional imaging study of “theory of mind” in story comprehension,” *Cognition*, vol. 57, no. 2, pp. 109 – 128, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/001002779500692R

[20] R. Prada and A. Paiva, “Intelligent virtual agents in collaborative scenarios,” 09 2005, pp. 317–328.

[21] Procedural Arts, “Façade,” Macintosh, Microsoft Windows, 2005.

[22] Maxis, “The Sims,” Microsoft Windows, Mac OS, OS X, Linux, PlayStation 2, GameCube, Xbox, 2000.

[23] M. H. Phan, J. R. Keebler, and B. S. Chaparro, “The development and validation of the game user experience satisfaction scale (guess),” *Human Factors*, vol. 58, no. 8, pp. 1217–1247, 2016, pMID: 27647156. [Online]. Available: https://doi.org/10.1177/0018720816669646

[24] F. Rheinberg, R. Vollmeyer, and S. Engeser, “Die erfassung des flow-erlebens,” *Diagnostik von Motivation und Selbstkonzept / hrsg. von Joachim Stiensmeier-Pelster und Falko Rheinberg. - Göttingen [u.a.] : Hogrefe, 2003. - (Tests und Trends ; N.F., 2). - ISBN 3-8017-1674-0. - S. 261 - 279*, 09 2007.

[25] P. M. F. Gomes, “MEEMOs: Believable Agents with Episodic Memory Retrieval,” Master’s thesis, Instituto Superior Técnico, 2010.

[26] N. Alea and S. Bluck, “Why are you telling me that? a conceptual model of the social function of autobiographical memory,” *Memory*, vol. 11, no. 2, pp. 165–178, 2003, pMID: 12820829. [Online]. Available: https://doi.org/10.1080/741938207

Code

Algorithm A.1: CiF-Ex Cycle pseudocode

```
begin
    if Not cycleCooldown() then
        if onGoingSEs >= maximumSEs or cycleCooldown or not PlayerNear then
            startCycleCooldown
            Return

        //DESIREFORMATION
        for cinCharacters do
            if c.busy or c.cooldown then
                Continue

            auxVolition ⟵ −INF
            auxSEName ⟵ " − "
            auxInitiatorName ⟵ " − "
            auxReceiverName ⟵ " − "
            for c2 in CharactersPlusPlayer do
                if c2.busy then
                    Continue
                for se in SocialExchanges do
                    se.initiator ⟵ c
                    se.receiver ⟵ c2
                    if se.initiatorVolition() > auxVolition then
                        auxVolition ⟵ se.initiatorVolition
                        auxSEName ⟵ se.name
                        auxInitiatorName ⟵ se.initiatorName
                        auxReceiverName ⟵ se.receiverName

            if auxVolition > mostWantedSE[c.name].volition then
                mostWantedSE[c.name].name ⟵ auxSEName
                mostWantedSE[c.name].initiatorName ⟵ auxInitiatorName
                mostWantedSE[c.name].receiverName ⟵ auxReceiverName
                mostWantedSE[c.name].volition ⟵ auxVolition

        for k in mostWantedSE do
            if mostWantedSE[k].volition > nextSE.volition then
                nextSE.name ⟵ auxSEName
                nextSE.initiatorName ⟵ mostWantedSE[k].initiatorName
                nextSE.receiverName ⟵ mostWantedSE[k].receiverName
                nextSE.volition ⟵ amostWantedSE[k].volition

        if nextSE.volition > 0 then
            getNPC(nextSE.initiatorName).startSE(nextSE.name, nextSE.receiverName, nextSE.volition)
        startCycleCooldown()
```

B

Tables and Figures

Table B.1: Social Exchanges in CiF-Ex

Term	**Function**
Social Exchange Name	SE Identifier
Initiator	The character that started the SE
Receiver	The character that was the target the SE
Other	An optional third participator
Intention	Keyword that defines the objective behind the SE
Initiator Influence Rules	Set of rules that return the initiator's volition
Receiver Influence Rules	Set of rules that determine the receiver's reaction
Pre-conditions	Rules that have to activate for the SE to happen
Outcome	How the receiver will respond to the SE
Effects	What will change when the SE concludes
Instantiations	The performance of the SE

Table B.2: Short description of the Social Exchanges

Social Exchange name	**Description**
Introduce	A character introduces themselves to another. If one of the participators is the Player, they learn the name of the NPC, instead of their nickname. This SE can only be performed once for each pair of characters
Compliment	The initiator says something pleasant about the receiver. Raises the Friendship SN
Insult	The initiator says something unpleasant about the receiver. Reduces the Friendship SN
Brag	The initiator says something impressive about themselves. Raises the Admiration SN
Flirt	The initiator tries to charm the receiver. Raises the Attraction SN
Small Talk	The initiator comments something about their day. Does not affect SN, but it is a nice and easy way to decrease some negative Status effects on the receiver
Ask For Food	The initiator asks for food to the receiver. If the receiver complies the initiator gains the food and their Hunger Physical Status decreases (in case of NPCs, the Player just gets food)
Ask For Drink	The initiator asks for a drink to the receiver. If the receiver complies the initiator gains the drink and their Thirst Physical Status decreases (in case of NPCs, the Player just gets drink)
Offer Food	Similar to the Ask For Food SE but with the giving and receiving roles inverted
Offer Drink	Similar to the Ask For Drink SE but with the giving and receiving roles inverted
Declare Love	The initiator asks the receiver to be a couple. If the receiver accepts the *Dating* relationship is set for the both of them
Date	The initiator asks the receiver on a date. This Social Exchange option is only available to couples
Break Up	The initiator will end their *Dating* relationship with the receiver. This Social Exchange option is only available to couples
Non Lethal Attack	The initiator strikes the receiver. The receiver has the change to strike back
Protect Other	The initiator calls out the receiver for either insulting or attacking another character. This SE is only available if the initiator has a memory of the receiver being hostile

Table B.3: Characters in CiF-Ex

Term	Function
Name	Unique Identifier
On Cooldown	Stops the character from starting SEs
Busy	Stops the characters from starting SEs with this character
Traits	Personality characteristics of the NPC
Status	Volatile attributes
Social Exchange Memory	Memories of all the SEs that they have perceived

Table B.4: Short description of the Traits

Trait name	Description
Friendly	The volition for Positive SEs is increased. The Social Exchange cooldown for this character will also be reduced
Charming	The volition for Romantic SEs is increased
Hostile	The volition for Negative SEs is increased
Shy	The volition for SEs in general is decreased. The Social Exchange cooldown for this character will also be increased
Gossiper	The volition for performing SEs in general is increased.
Obnoxious	The character will be more compelled to repeat SEs that they have just performed
Humble	The volition for Negative SEs is decreased along with the Brag Social Exchange
Aggressive	The volition for Negative SEs is increased
Coward	The initiator volition for SEs in general is decreased
Brave	The volition for the Ask For Food/Drink and Protect Other is increased
Faithful	If they are in a relationship, the volition for Romantic SEs with other participators that aren't their partner is decreased
Unfaithful	The volition for Romantic SEs is increased
Gender	Whether the NPC is male or female

Table B.5: *CIF_SocialExchange_Component*

Element name	**Implementation**
Social Exchange Name	String
Initiator Name	String
Receiver Name	String
Other Name	String (optional)
Intention	Enum_CIF_SE_Intention
Weights	Array of Floats
Initiator Volition	Float
Receiver Volition	Float
Pre-conditions	Function that returns a volition (float) or aborts the SE process
Microtheories	Collection of functions that return a volition (float)
Volition Function	Function that calculates the initiator or receiver volition
Assert Outcome	Function that calculates the outcome
Outcome	Integer
Select Animation	Returns an ECharacter_Emote
Execute Social Changes	Function that modifies the Social State and characters' Status

Table B.6: *Struct_CIF_SE_Attributes*

Element name	**Implementation**
Intention	Enum_CIF_SE_Intention
UpThreshold	Float
DownThreshold	Float
Present Continuous	String
Positive Reaction	String
Neutral Reaction	String
Negative Reaction	String
Initiator Animation	ECharacter_Emote
Receiver Positive Animation	ECharacter_Emote
Receiver Neutral Animation	ECharacter_Emote
Receiver Negative Animation	ECharacter_Emote

Table B.7: *W_CIF_NPCDialogue*

Element name	**Implementation**
NPC Name	String
NPC Known Name	String, can be the *Nickname* or the *NPC Name*
Social Exchange	Enum_CIF_Social_Exchanges
Dialogue Line	String, next line to display
Lines to Display	String Array
NPC Started?	Boolean, `True` if the NPC started the interaction
Replace All *	Function that modifies and returns a String
Check Special Characters	Function that identifies special characters in a String
Count Consumables	Function that counts the consumables in the Player's inventory

Table B.8: *BP_CIF_NPC*

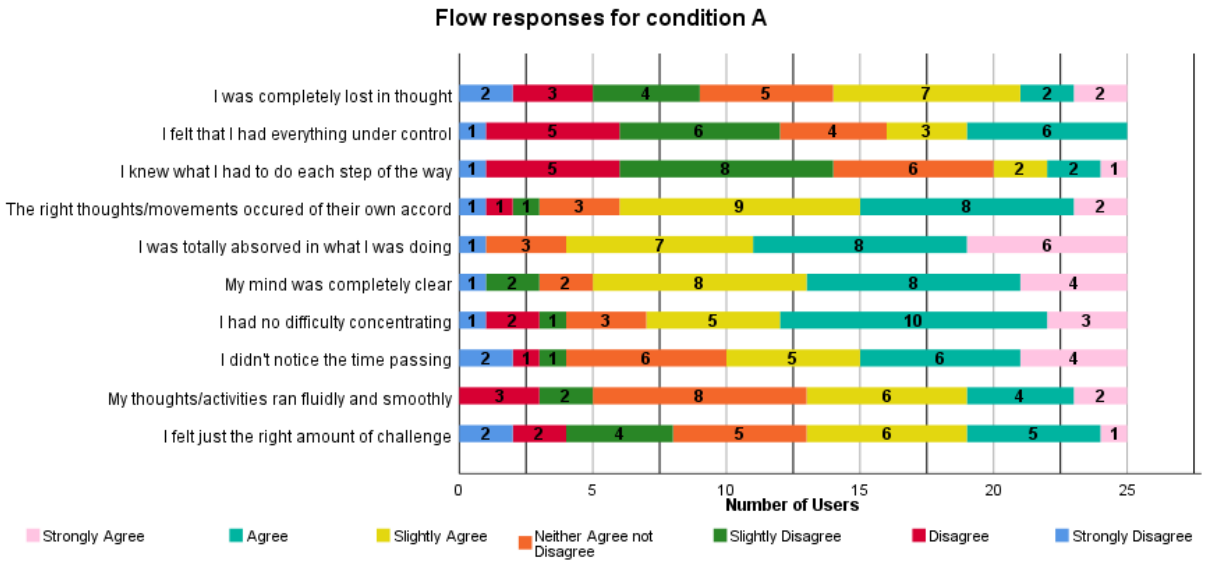
Element name	**Implementation**
Name	String
Nickname	String
CiF ID	Unique ID
CiF Ready?	Boolean, `True` if the NPC can start a SE
Having Interaction?	Boolean, `True` if the NPC is part of an ongoing SE
Social Attributes	Struct_CIF_Social_Attributes
Social Exchange Memory	Array of Struct_SE_Memory
Friendship SN Beliefs	Struct_CIF_SN_Beliefs
Admiration SN Beliefs	Struct_CIF_SN_Beliefs
Attraction SN Beliefs	Struct_CIF_SN_Beliefs
Home Location	Vector (world position)
CiF Behaviour Tree	BT_CIF_Dialog
CiF Dialogue Component	CIF_Dialogue_Component
Social Exchange	CIF_SocialExchange_Component
Resting?	Boolean, `True` when the NPC is resting
Go Rest	Event that starts the resting behaviour
Walk to Object	Event that makes the NPC walk to a game object
Wander	Event that starts the wandering behaviour
Init Social Exchange	Event that starts the SE

Table B.9: *Struct_CIF_Social_Attributes*

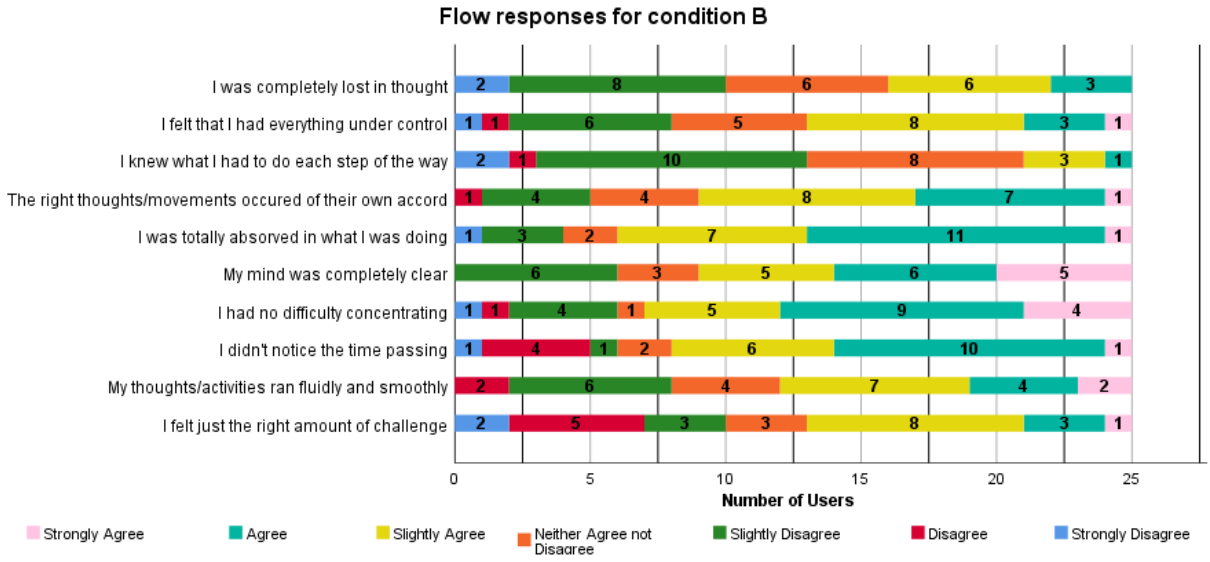
Element name	**Implementation**
Friendship Social Network	*Struct_CIF_Social_Network*
Attraction Social Network	*Struct_CIF_Social_Network*
Admiration Social Network	*Struct_CIF_Social_Network*
Social Emotion Status	*Struct_CIF_Social_Status*
Emotion Status	*Struct_CIF_Emotion_Status*
Physical Status	*Struct_CIF_Physical_Status*
Traits	*Struct_CIF_Traits*

Table B.10: *BP_CIF_GameState*

Element name	**Implementation**
Save Data	CIF_Save
Next SE	Struct_CIF_Social_Exchange_Volition
Most Wanted SE	Map of String to Struct_CIF_Social_Exchange_Volition
Social Exchange Classes	Map of String to CIF_Social_Exchange_Component class
Ongoing SEs	Array of Struct_CIF_Social_Exchange_Volition
Bonfire	BP_PL_CIF_CraftingBonfire
Player Interacting?	Boolean, `True` when the Player is interacting with a character
Dead NPCs	Array of Strings
Calculate Volitions	Function that starts the volition calculation
Calculate Next SE	Function selects the highest volition SE
Start Most Wanted SE	Event that starts the SE with highest volition
Player Attack	Event that makes the Player attack
Save Game	Event that saves the game
Load Save	Event that loads the game
Social Exchange	*CIF_SocialExchange_Component*

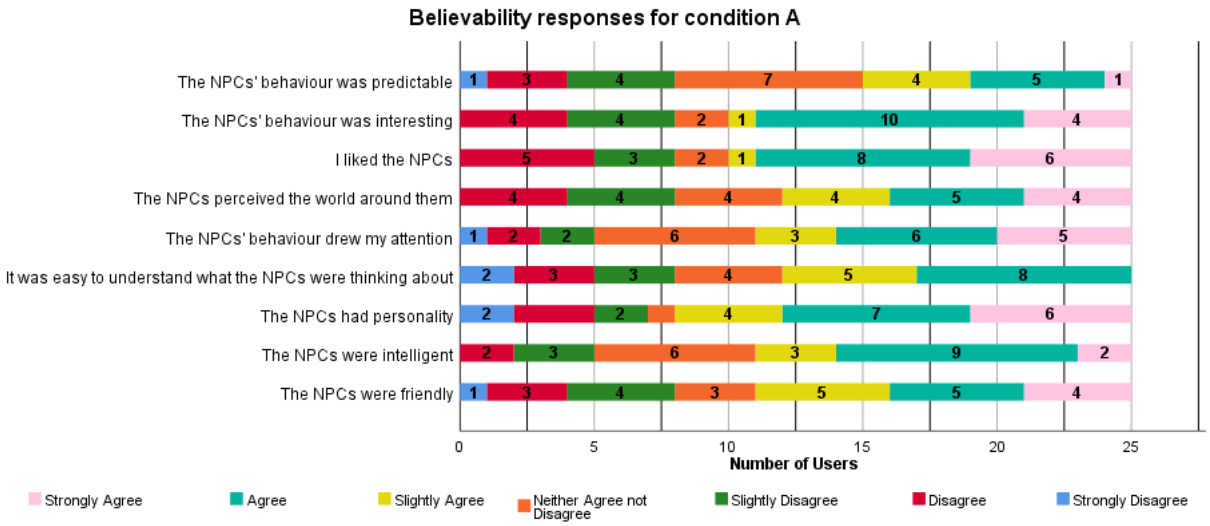


(a) Stacked chart of answers for condition A's Flow

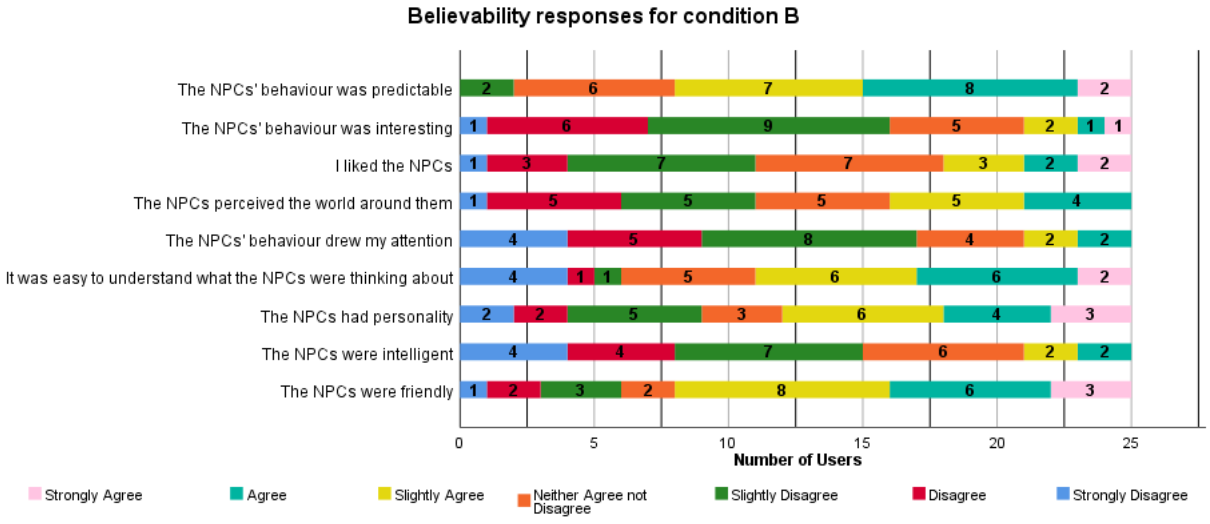


(b) Stacked chart of answers for condition B's Flow

Figure B.1: Stacked chart of answers for Flow

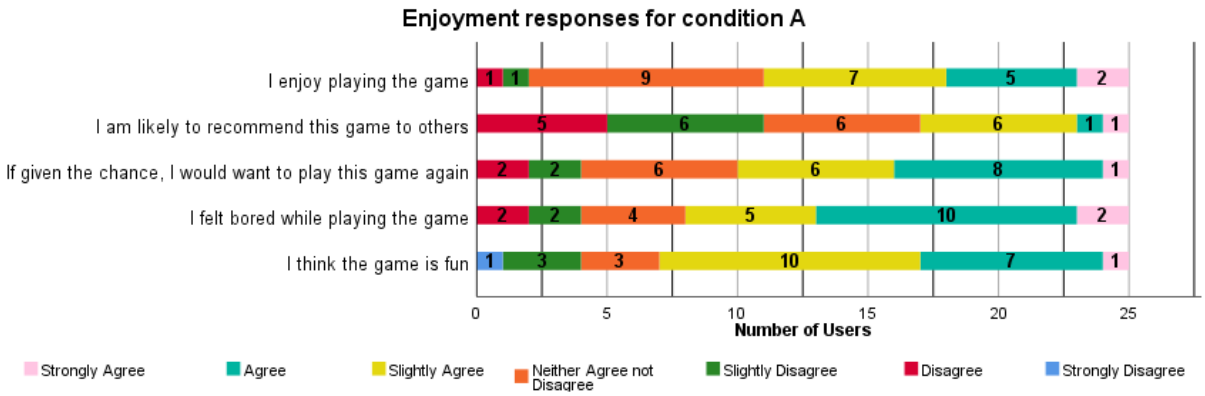


(a) Stacked chart of answers for condition A's Believability

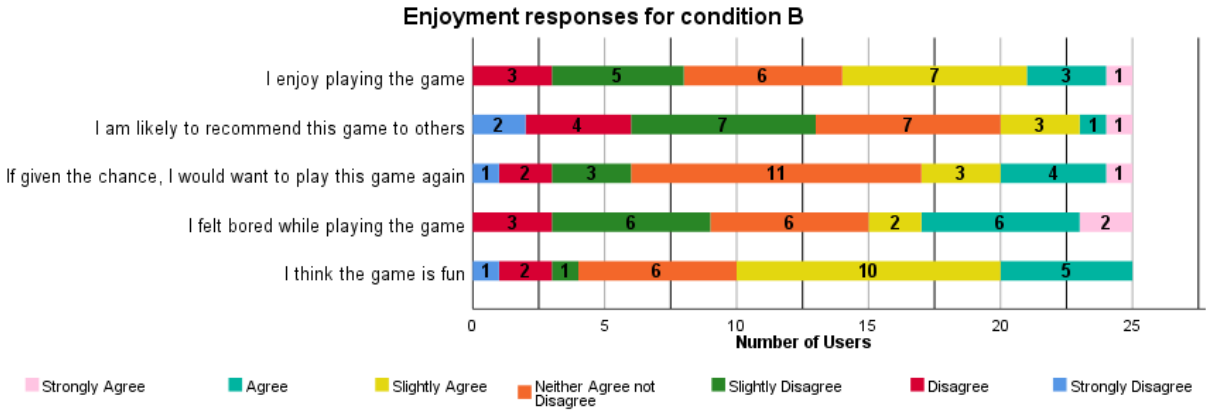


(b) Stacked chart of answers for condition B's Believability

Figure B.2: Stacked chart of answers for Believability



(a) Stacked chart of answers for condition A's Enjoyment



(b) Stacked chart of answers for condition B's Enjoyment

Figure B.3: Stacked chart of answers for Enjoyment

SE	Phase	recondition/Othe	SocialNetwork								
			Attraction			Friendship			Admiration		
			Goal	Current	Belief	Goal	Current	Belief	Goal	Current	Belief
Introduce	iVolition	NotIntroduced?	+0.5	+0.5	+0.5				+0.5	+0.5	+0.5
	rVolition	NotIntroduced?	+0.5	+0.5	+0.5				+0.5	+0.5	+0.5
	PositiveOutcome	Know Name									
	NeutralOutcome	Know Name									
Introduce	NegativeOutcome	Know Name									
Compliment	iVolition		+1	+1		+1				+1	
	rVolition		+1	+1		+1.5	+0.5			+1	
	PositiveOutcome					+1	+1	+1			
	NeutralOutcome										
Compliment	NegativeOutcome					-1	-1	-1	-1	-1	-1
Brag	iVolition		+1	+0.5			+0.5				se positivo
	rVolition			+1			+0.5			+0.5	
	PositiveOutcome			+1	+1					+1	+1
	NeutralOutcome										
Brag	NegativeOutcome			-1				-1		-1	-1
SmallTalk	iVolition		+0.5	+0.5	+0.5	+0.5	+0.5	+0.5			
	rVolition		+0.5	+0.5	+0.5	+0.5	+0.5	+0.5	+0.5	+0.5	+0.5
	PositiveOutcome										
	NeutralOutcome										
SmallTalk	NegativeOutcome										
ProtectOther	iVolition	+other's affectiveness		-1			-1			-1	
	rVolition			+1			+1			+1	
	PositiveOutcome										
	NeutralOutcome							-1			
ProtectOther	NegativeOutcome						-1	-1			
AskForFood	iVolition	iv: -20					+1				
	rVolition			+1		+1	+2	+1		+1	
	PositiveOutcome					+1	+1	+1	+1	+1	
	NeutralOutcome							-1			-1
AskForFood	NegativeOutcome					-1	-1	-1	-1	-1	-1
AskForDrink	iVolition	iv: -20					+1				
	rVolition			+1		+1	+2	+1		+1	
	PositiveOutcome					+1	+1	+1	+1	+1	
	NeutralOutcome										
AskForDrink	NegativeOutcome					-1	-1		-1	-1	
OfferFood	iVolition	iv: -30		+0.5		+0,5	+1			+1.5	
	rVolition			+1		+1	+2	+1		+1	
	PositiveOutcome					+1	+1	+1	+1	+1	
	NeutralOutcome							-1			-1
OfferFood	NegativeOutcome					-1	-1	-1	-1	-1	-1

Figure B.4: Example of Weights for the Influence Rules. The values in blue are only applied to the initiator, the values in red are only applied to the receiver and the values in black are for the both of them