David Cristóvão Ricardo

Friendly Lords

Implementing an Artificial Intelligence Social Architecture in Mount & Blade II: Bannerlord

UNIVERSIDADE DO ALGARVE

Faculdade de Ciências e Tecnologia

2021

David Cristóvão Ricardo

Friendly Lords

Implementing an Artificial Intelligence Social Architecture in Mount & Blade II: Bannerlord

Master in Informatics Engineering

Trabalho efetuado sob a orientação de:
Prof. João Miguel de Sousa de Assis Dias

UNIVERSIDADE DO ALGARVE

Faculdade de Ciências e Tecnologia

2021

Friendly Lords

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

____

David Cristóvão Ricardo

Acknowledgments

I would like to thank my family and my parents for their friendship and encouragement over all these years, for always being there for me.

I would also like to acknowledge my dissertation supervisor Prof. João Dias, and Prof. Pedro Santos for their support and sharing of knowledge that has made this Thesis possible.

To each and every one of you – Thank you.

Abstract

Despite living in a golden age for video games where there is an increase in the development and popularity of new technologies, such as Virtual Reality and Augmented Reality, and constant improvements being made on immersion, either through rich storytelling, high graphical fidelity or even gameplay itself, one area that is generally lacking on innovation is the social interaction of Non-Playable Characters.

The credibility of these virtual characters (usually called NPCs) requires that they have human characteristics, such as emotions and the ability to think and make decisions depending on their own will. One of the most important characteristics is the ability to socialize and interact with each other.

The main objective of this work was to implement a Social Architecture of Artificial Intelligence in the game Mount&Blade II: Bannerlord to enhance the credibility of NPCs, making them socially more active and interesting, and improving the User Experience. In the game in which this study takes place (respectively Mount&Blade II: Bannerlord), although there are plenty of NPCs in the world, these are extremely limited, predictable and rarely demonstrate social behaviors.

It also addresses some socio-emotional architectures such as CiF and FAtiMA. While FAtiMA is more focused on the generation of emotions, and how those emotions may affect the behaviour of characters, CiF has an explicit representation of social relations between NPCs and how the can influence behaviour. Due to this, the system architecture used as the basis for our model is the CiF architecture, also used before in some popular games like Skyrim and Conan Exiles. It is a system designed to be able to generate social behavior for the social agents, in this case the virtual characters. Instead of looking static and appearing to be clones of each other, the NPCs will appear more natural, making them more interesting and believable from the point of view of social interaction.

One of the major differences between the CiF model and the developed model, named Comme il Faut - Bannerlord (CiF-BL), was, respectively, an addition of a centralized component, which will manage about which NPCs will be able to engage in interactions, how many Social Interactions will be able to take place at the same time, when these are started and finished.

From CiF-BL's perspective, the player is just another character, and just as NPCs will want to interact with each other, they will also want to interact with the player.

To do this work, it was necessary to adapt the game's existing dialog system to make CiF-BL's changes possible. Adding the options that allow the player to interact socially with an NPC and vice-versa. It was also necessary to adapt and implement another game system, responsible for signaling and informing the most relevant locations and characters. This adaptation was responsible for making it possible to visualize the interactions between the different NPCs.

Thus, during the time that the user is present in the virtual world, the virtual characters will no longer have a totally irrelevant and figurative role in the game, and will have a greater social participation. With the goal of appearing more believable, natural, and, to look more like "living beings".

The developed model was validated and evaluated through user tests. Unfortunately, due to COVID-19 restrictions that occurred during the execution of this thesis, it was not possible to conduct a controlled in-person evaluation (which would be the ideal way to evaluate our work). Instead, a mod was produced and released online to the Bannerlords player and modding community. The only difference between the mod and the base version of the game was the CIF-BL model used to control the NPCs. The mod was called "Friendly Lords" and released in 18th of August 2021, both in NexusMods and in ModDB. The mod was announced on social networks, such as Reddit, in groups that were related to the game.

Participants were voluntarily invited to complete an anonymous questionnaire regarding their gaming experience, and the answers were collected. From the analysis of these same answers, very favorable results were obtained for the objectives we defined.

However, overall, the feedback from players and the community has been very positive about the mod and its modifications to the game. Some examples are that it looks promising, that they want active development, and that are grateful to help bring new life to the game. People give suggestions and constructive feedback to implement in the mod and help get a better experience. Meanwhile, some players have shown more interest and offered to collaborate directly in the development, either for more dialogue options between the NPCs, or to translate the mod respectively into Turkish which is the native language of the studio currently developing the game.

In short, using the social architecture implemented in the game, the Credibility of NPCs and the Social Presence improved by more than 30%, having successfully achieved the goals.

This document describes in more detail the process of researching, implementing and testing the model. To improve this work and the Artificial Intelligence social architecture, it would be necessary to at least add more personality traits and a greater number of different social

interactions, for the NPCs to have a greater diversity in terms of their social behavior.

Keywords

Social Architecture; Non-Playable Character; Social Interaction; Artificial Intelligence.

Resumo

Apesar de vivermos numa era dourada para os videojogos onde há um aumento do desenvolvimento e da popularidade de novas tecnologias, como a Realidade Virtual e a Realidade Aumentada, e constantes melhorias a serem feitas na imersão, seja através de ricas narrativas, fidelidade gráfica elevada ou até mesmo gameplay em si, uma área que geralmente não é alvo de inovação é a interação social entre as Personagens Não Jogáveis.

A credibilidade dessas personagens virtuais (normalmente chamadas de NPCs) exige que estas tenham características humanas, como emoções e a capacidade de pensar e tomar decisões dependendo da sua própria vontade. Uma das características mais importante é a capacidade de socializar e interagir uns com os outros.

O objectivo principal deste trabalho é implementar uma Arquitectura Social de Inteligência Artificial no jogo Mount&Blade II: Bannerlord para incrementar a Credibilidade dos NPCs, tornando-os socialmente mais ativos e interessantes, e melhorar a Experiência do Utilizador. No jogo em que se realiza este estudo (respetivamente Mount&Blade II: Bannerlord), embora haja uma abundância de NPCs no mundo, estes são extremamente limitados, previsíveis e raramente demonstram comportamentos sociais.

Este trabalho aborda diferentes arquitecturas socio-emocionais, como Comme il Faut (CiF) e FAtiMA. Enquanto FAtiMA está mais centrado na geração de emoções, e como essas emoções podem afectar o comportamento das personagens, CiF tem uma representação explícita das relações sociais entre os NPCs e de que forma o comportamento pode influenciar. Devido a isto, a arquitetura do sistema usada como base para o nosso modelo é a arquitetura CiF, também já usada antes em alguns jogos populares como Skyrim e Conan Exiles. É um sistema concebido para ser capaz de gerar comportamentos sociais para os agentes sociais, neste caso, os personagens virtuais. Em vez de parecerem estáticos e aparentarem serem clones uns dos outros, os NPCs vão parecer mais naturais, tornando-os mais interessantes e credíveis do ponto de vista da interação social.

Uma das maiores diferenças entre o modelo CiF e o modelo final desenvolvido, nomeado de Comme il Faut - Bannerlord (CiF-BL), foi, respectivamente, uma adição de um componente centralizado, que irá fazer a gestão acerca de quais os NPCs que estarão aptos para se envolverem

em interações, quantas Interações Sociais serão capazes de se realizar ao mesmo tempo, quando estas são iniciadas e terminadas. Da perspectiva do CiF-BL, o jogador é apenas uma outra personagem e, assim como os NPCs vão querer interagir uns com os outros, eles também vão querer interagir com o jogador.

Para realizar este trabalho, foi necessário adaptar o atual sistema de diálogo já existente do jogo para possibilitar as alterações do CiF-BL. Adicionando assim as opções que permitem ao jogador interagir socialmente com um NPC e vice-versa. Foi necessária também uma adaptação e a implementação de um outro sistema do jogo, responsável por assinalar e informar as localizações e os personagens mais relevantes. Esta adaptação foi responsável para tornar possível a visualização das interações entre os diferentes NPCs.

Assim, durante o tempo em que o utilizador estiver presente no mundo virtual, as personagens virtuais irão deixar de ter um papel totalmente irrelevante e figurativo no jogo, e, terão uma maior participação social. Com o objectivo de parecerem mais credíveis, naturais, e, de se assemelharem mais com "seres vivos".

Para a validação e a avaliação do modelo, este foi sujeito a testes de utilizador. Infelizmente, devido às restrições COVID-19 que ocorreram durante a execução desta tese, não foi possível realizar uma avaliação controlada presencialmente (que seria a forma ideal de avaliar o nosso trabalho). Em vez disso, um mod foi produzido e lançado online para os jogadores do jogo Bannerlord e para a comunidade modding. A única diferença entre o mod e a versão base do jogo foi o modelo CIF-BL usado para controlar e melhorar os NPCs. O mod foi chamado de "Friendly Lords" e lançado em 18 de agosto de 2021, tanto no site NexusMods como no ModDB. O mod foi anunciado nas redes sociais, como o Reddit, em grupos que estavam relacionados com o jogo.

Os participantes foram convidados voluntariamente a preencher um questionário anónimo relativamente à sua experiência de jogo, sendo feita a recolha das respostas. A partir da análise dessas mesmas respostas, foram obtidos resultados muito favoráveis para os objectivos que definimos.

Entretanto, no geral, o feedback dos jogadores e da comunidade foi muito positivo sobre o mod e sobre as suas modificações no jogo. Alguns exemplos são que parece promissor, que querem um desenvolvimento ativo, e que estão gratos por ajudar a trazer uma nova vida ao jogo. As pessoas dão sugestões e feedback construtivo para implementar no mod e ajudar a obter uma melhor experiência. Entretanto, alguns jogadores mostraram mais interesse e disponibilizaram-se para colaborar diretamente no desenvolvimento, quer seja para mais opções de diálogos entre os NPCs, quer seja para a tradução do mod, respectivamente para Turco que é a língua materna do estúdio que atualmente desenvolve o jogo.

Resumindo, usando a arquitetura social implementada no jogo, a Credibilidade dos NPCs e a qualidade do comportamento social melhorou mais de 30%, tendo atingido com sucesso os objetivos. Este documento descreve mais em pormenor o processo de pesquisa, de implementação e de teste do modelo. Para melhorar este trabalho e a sua arquitetura social de Inteligência Artificial, seria necessário, pelo menos, adicionar mais traços de personalidade e um maior número de diferentes interações sociais, para os NPCs terem uma maior diversidade quanto ao seu comportamento social.

Palavras Chave

Arquitectura Social; Personagem Não-Jogável; Interação Social; Inteligência Artificial.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ActAffAct	Acting Affectively affecting Acting
CiF	Comme il Faut
EE	Ensemble Engine
FAtiMA	Fearnot AffecTIve Mind Architecture
FLAME	Fuzzy Logic Adaptive Model of Emotions
MT	Microtheory
NPC	Non-Playable Character
ORIENT	Overcoming Refugee Integration with Empathic Novel Technology
SE	Social Exchange
UI	User Interface

1 Introduction

Contents

Despite living in a golden age for video games where there are constant improvements being made on immersion, either through rich storytelling, high graphical fidelity or even gameplay itself, one area that is generally lacking on innovation is the social interaction of Non-Playable Character (NPC) [16]. The social exchanges amongst the different NPCs, both between the player and between themselves, still remain relatively basic. Normally their dialogue is repetitive and the NPCs who doesn't contribute to the main story only are present on to 'fill' the game settings, like villages, towns, etc. As an example, in the popular game series of Ubisoft, named Assassin's Creed, there are many characters that do not have any specific influence on the main story, or even a dialogue with the player or with other characters.

1.1 Objectives

We believe that improving the social interaction between NPCs will help make games more engaging and memorable. This effect might be particularly noticeable in open-world games that are populated with large numbers of NPCs. One example of such a game is the AAA video game Mound&Blade II: Bannerlord, a strategy action role-playing game, where the player can join current factions, battle as a mercenary, be an outlaw or remain neutral. The game will be described in more detail in Chapter 3.

In this video game, although NPCs abound throughout the world, they are very limited and seldom exhibit social behaviour. There are taverns in the cities spread out along the world of Calradia, but the existing NPCs do not appear to have a life of their own, as they do not interact with each other, and most of them barely interact with the player for that matter. The NPCs that interact the most with the player are the quest givers and nobles, and most of that interaction is instrumental rather than social. For instance, if the player completes a quest for a quest giver, it will be rewarded with additional troops for recruitment.

Therefore, in this thesis, we aim to improve the NPCs and Player interactions in Bannerlord, by focusing in creating more socially interesting and believable NPCs, and thus improving the player's experience.

Additionally, we want to achieve this without a costly authoring effort. Creating more believable and engaging NPCs can be achieved by creating detailed characters by scripting a more detailed set of rules for each character. However, given the usually large number of NPCs in Bannerlord, we want to achieve this goal with a smaller scripting effort. To that end, we can draw ideas from an area of research that is concerned with the creation of autonomous agents that are able to exhibit socio-emotional behavior.

These type of systems, called socio-emotional agent architectures, focus on the creation of agents that act and make decisions based on desires, emotions and interpersonal relations. FAtiMA [2] and CIF [17] are examples of socio-emotional agent architectures. These architectures will allow us to endow NPCs with behavior based on desires and interpersonal relations, having different personalities and behaviors. Instead of appearing static and looking as clones of each other, NPCs will look more natural, making them more interesting and more believable from a social interaction point of view.

As such, in this work, we will explore the use of social-emotional agent architecture (specifically the original CIF - Comme il Faut) and its integration with the game. Also, the study of how the CiF model improves the social engagement and user experience of users playing the game.

The main contributions of the work are the CiF model extention, implementation and integration into the Bannerlords' game. The CiF model developed will be named CiF-BL.

1.2 Organization of the Document

This thesis is organized as follows:

In the next chapter, we present the related work. We start by introducing relevant concepts, we then provide an overview of existing social agent architectures, and describe games that explore social interaction. We end the chapter with a discussion on what elements could be relevant for our goal. Chapter 3, introduces the game Bannerlord, and provides an overview of the architecture behind Bannerlord's modding system. In Chapter 4, we present and discuss the proposed model for our solution, which is based on the CIF architecture. Afterwards, in, Chapter 5 we provide a more detailed description of how the model was implemented and integrated into the game engine to create a new mod for Bannerlord: Friendly Lords. Chapter 6 describes how we evaluated the mod, and presents the results and feedback obtained from the evaluation with users. Finally, we end with some conclusions about the performed work and results, and discuss future work.

2

Related Work

Contents

In this chapter, we provide a description of important concepts related to our goal, related work on existing social agent architectures, and finally we present a set of games where socially intelligent agents have been applied.

2.1 Background

2.1.1 Towards believable NPCs

In the previous chapter, we highlighted the importance of creating NPCs that are perceived as more natural and social, but we first need to clarify what we understand as an NPC. NPC, which means Non-Playable Character, is any character that is not controlled by the player, but it is controlled by the game, and its behavior is usually scripted, triggered only by certain event on environment, action or dialog with the player avatar.

Another important concept is the concept of believability. NPC Believability refers to how much they are believable from an observer's perspective - the extent to which an observer is willing to suspend his disbelief and consider the NPC as a real person or character. According to Thomas and Johnson, believability provides the illusion of life [18]. A character is also considered believable if someone believes that it is a player controlling the other character, or if the character is an actual living being [19].

In addition to believability, personality, and also, emotions are important components for the creation of NPCs [20]. Normally, NPCs who are believable must have emotions to express their feelings to the user. This is also known as an Emotional State, that must be linked to the reasoning process to determinate the actions to take to reach their objectives.

The Personality is an important component because it makes all the NPCs different from each other, more friendly or unfriendly, more curious or afraid. It determines goals and desires, and are those who act and interact with the players or with the other NPCs, as if they were other players, increasing the user experience. For example, two NPCs may act differently on the same situation because of their personality [21].

2.1.2 Modding

A mod or modification of a computer game works by changing the content from the video game in order to make it operate in a different way from its original. In games that support modding, they can be usually be made by non-programmers using the tools provided by the game, and in some cases, it is even possible to transform an existing game into a completely new game.

Depending on who you ask or where you look, game mods mean different things to different people. They can include new items, weapons, characters and enemies, textures, levels and maps, storylines, music and sounds, money, armor, life, and game modes, also they can be single-player or multiplayer [16, 22].

The Internet provides is a free medium to promote and distribute mods, consequently, they have become an increasingly important factor in the success of some games. Popular websites dedicated to modding include NexusMods [1] and Mod DB [2] and where users can either publish, release and request mods.

One of the reasons for the selection of Bannerlord for this thesis is that it provides strong support for modding, even allowing programmers indirect access to the source code of the game, and the capability of writing new mods as c# modules that can be integrated into the game. Furthermore, Bannerlord has also an enormous and active modding community and forums supported by TaleWorlds. Bannerlord and its architecture will be discussed in more detail in Chapter 3.

2.2 Social Architectures and Applications

2.2.1 ActAffAct

Acting Affectively affecting Acting (ActAffAct) is an emotional agent architecture designed to create believable virtual characters for games or other type of interactive systems [23]. These agents must be capable to remember their goals, and they should be able to act, perceive, expect and anticipate.

The system operating is an action-reaction loop that can be described in four phases:

- It starts with the perception of the environment, converts the external information into a comprehensible form for cognitive appraisal of the agent;

- Appraisal. After getting the information from the environment, this will check the appraisal criteria, also known as Stimulus Evaluation Checks and Appraisal Registry, which causes several response tendencies. Every perception passes through appraisal, and, emotions are generated based on appraisal, and the agents can "feel" hope, fear, and joy, for example, with one valuable intensity;

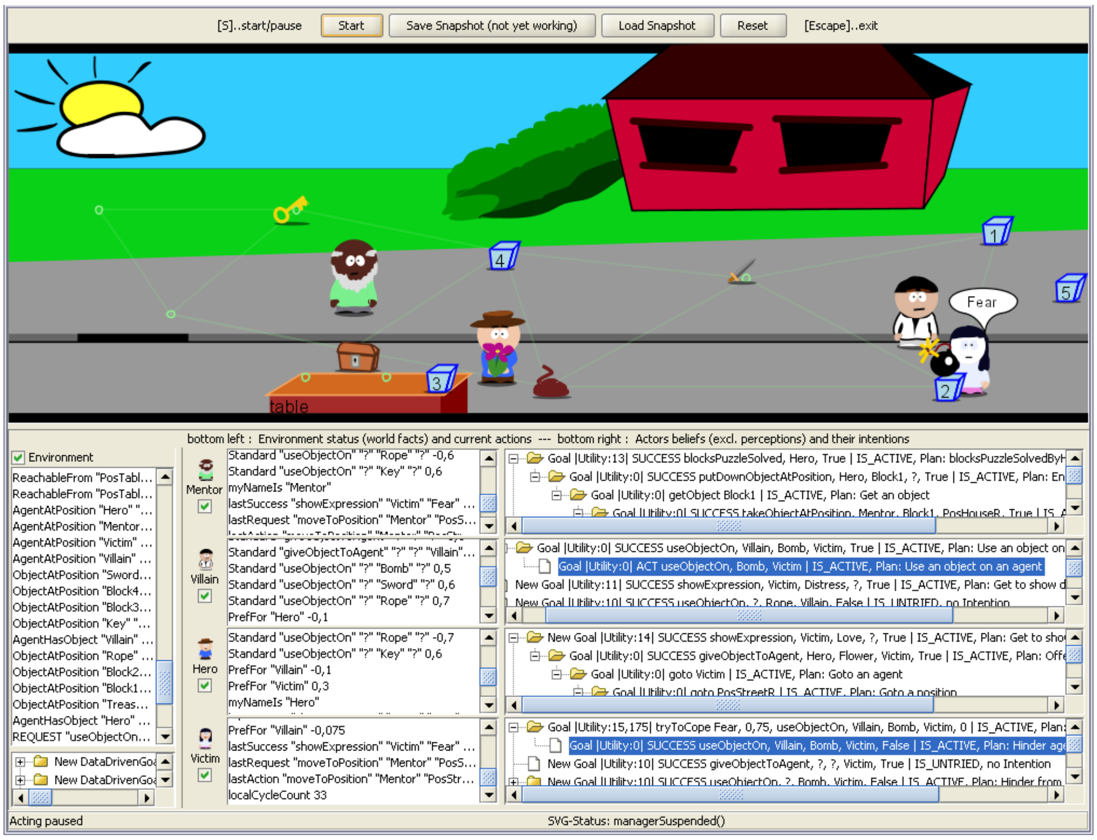
[1]NexusMods, https://www.nexusmods.com
[2]ModBD, https://www.moddb.com

- Decision-Making. Depending on the goal significance or relevance and responsibility of the agent, to be considered by the Action/Execution part of the model.

  For example, in a simple case where exist goals to either try to prevent or to encourage an action to happen. If the perception which declares the completion of this action is encountered, a goal encouraging it would lead to a positive relevance for this goal, preventing a negative one.

  The RAT (Relational Action Tendency) tries to reconfigure the currently executed actions and behaviors based on experienced emotions. If the perception is positive, it must increase the intensity of an appraisal, if it is negative it must decrease;

- Execution: after the agent takes the Action, this affects the environment and the loop stills going [1].



```
bottom left : Environment status (world facts) and current actions --- bottom right : Actors beliefs (excl. perceptions) and their intentions

ReachableFrom "PosTabl...
ReachableFrom "PosTabl...
AgentAtPosition "Hero" "...
AgentAtPosition "Mentor...
AgentAtPosition "Victim" ...
AgentAtPosition "Villain" ...
ObjectAtPosition "Sword...
ObjectAtPosition "Block4...
ObjectAtPosition "Block3...
ObjectAtPosition "Key" "...
AgentHasObject "Villain" ...
ObjectAtPosition "Rope" ...
ObjectAtPosition "Block2...
ObjectAtPosition "Block1...
ObjectAtPosition "Treas...
AgentHasObject "Hero" ...
REQUEST "useObjectOn...

New DataDrivenGoa
New DataDrivenGoa

Mentor:
Standard "useObjectOn" "?" "Rope" "?" -0,6
Standard "useObjectOn" "?" "Key" "?" 0,6
myNameIs "Mentor"
lastSuccess "showExpression" "Victim" "Fear" ...
lastRequest "moveToPosition" "Mentor" "PosS...

Villain:
Standard "giveObjectToAgent" "?" "?" "Villain"...
Standard "useObjectOn" "?" "Bomb" "?" 0,5
Standard "useObjectOn" "?" "Sword" "?" 0,6
Standard "useObjectOn" "?" "Rope" "?" 0,7
PrefFor "Hero" -0,1

Hero:
Standard "useObjectOn" "?" "Rope" "?" -0,7
Standard "useObjectOn" "?" "Key" "?" 0,6
PrefFor "Villain" -0,1
PrefFor "Victim" 0,3
myNameIs "Hero"

Victim:
PrefFor "Villain" -0,075
lastSuccess "showExpression" "Victim" "Fear" ...
lastRequest "moveToPosition" "Mentor" "PosS...
lastAction "moveToPosition" "Mentor" "PosStr...
localCycleCount 33

Goal |Utility:13| SUCCESS blocksPuzzleSolved, Hero, True | IS_ACTIVE, Plan: blocksPuzzleSolvedByH
  Goal |Utility:0| SUCCESS putDownObjectAtPosition, Hero, Block1, ?, True | IS_ACTIVE, Plan: En
    Goal |Utility:0| getObject Block1 | IS_ACTIVE, Plan: Get an object
      Goal |Utility:0| SUCCESS takeObjectAtPosition, Mentor, Block1, PosHouseR, True | IS_A

Goal |Utility:0| SUCCESS useObjectOn, Villain, Bomb, Victim, True | IS_ACTIVE, Plan: Use an object on
  Goal |Utility:0| ACT useObjectOn, Bomb, Victim | IS_ACTIVE, Plan: Use an object on an agent
New Goal |Utility:11| SUCCESS showExpression, Victim, Distress, ?, True | IS_ACTIVE, Plan: Get to show d
New Goal |Utility:10| SUCCESS useObjectOn, ?, Rope, Villain, False | IS_UNTRIED, no Intention

New Goal |Utility:14| SUCCESS showExpression, Victim, Love, ?, True | IS_ACTIVE, Plan: Get to sho
  Goal |Utility:0| SUCCESS giveObjectToAgent, Hero, Flower, Victim, True | IS_ACTIVE, Plan: Offe
    Goal |Utility:0| goto Victim | IS_ACTIVE, Plan: Goto an agent
      Goal |Utility:0| goto PosStreetR | IS_ACTIVE, Plan: Goto a position

Goal |Utility:15,175| tryToCope Fear, 0,75, useObjectOn, Villain, Bomb, Victim, 0 | IS_ACTIVE, Plan:
  Goal |Utility:0| SUCCESS useObjectOn, Villain, Bomb, Victim, False | IS_ACTIVE, Plan: Hinder age
New Goal |Utility:10| SUCCESS giveObjectToAgent, ?, ?, Victim, True | IS_UNTRIED, no Intention
New Goal |Utility:10| SUCCESS useObjectOn, ?, Bomb, Victim, False | IS_ACTIVE, Plan: Hinder from

Acting paused                    SVG-Status: managerSuspended()
```

Figure 2.1: ActAffAct Interface [1].

ActAffAct was used by a basic scenario with multiple virtual characters. These characters, also treated as agents, in accordance with the knowledge of previous emotions, choose the suitable target, and then a proper response [1].

2.2.2 FAtiMA

Fearnot AffecTIve Mind Architecture (FAtiMA) is an Agent Architecture with planning capabilities designed to use emotions and personality to influence the agent's behavior [2].

In the previous years, the architecture was used in multiple scenarios (for example FearNot!, ORIENT, and a process Model of Empathy [24]) and by some research institutions, which led to the architecture being added with several new attributes.

FAtiMA is composed by the FAtiMA Core that defines how the agents' architecture works, and the Modular components that can be added. The FAtiMA phases of the processing cycle

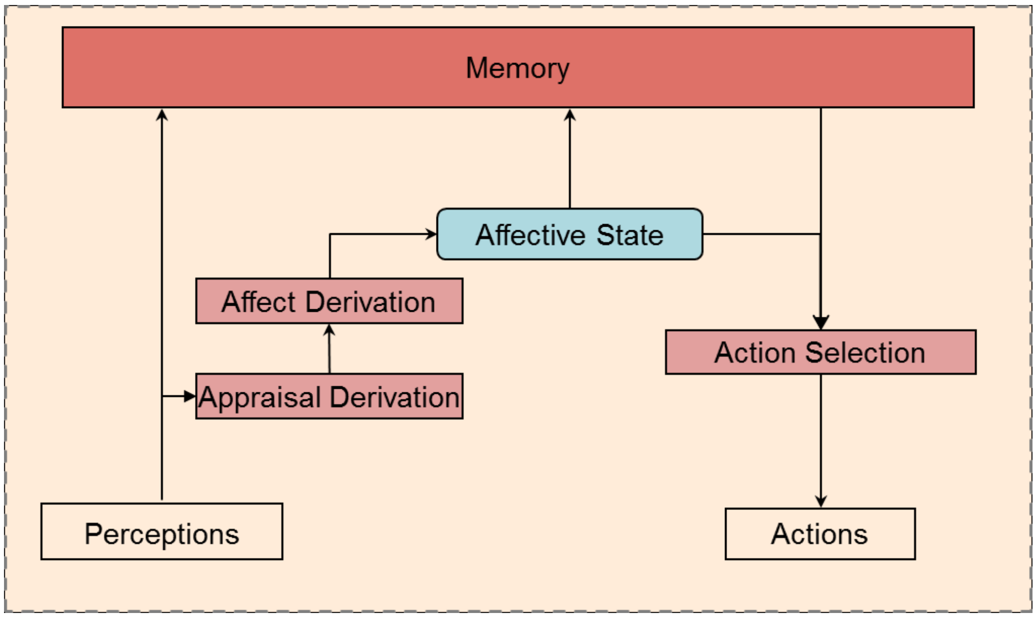


Figure 2.2: FAtiMA Core Architecture [2].

may be enumerated as:

- **Perceptions**, each agent can perceive events from the environment using their own sensors, being stored and used to update the agent's memory;

- Appraisal, the agent appraises the perception's significance and triggers the appropriate emotions.

  This process can be divided into two different ones:

  - the **Appraisal Derivation**, responsible for evaluating the relevance of the event to the agent and determining a set of appraisal variables, and,
  - the **Affect Derivation**, taking the appraisal variables and generating as output and being stored in the Affective State (emotions and/or mood).

Thus, the output of the Appraisal process is stored in the Affective State.

- After this update on the agent's **Affective State**, with the previous information stored in memory about previous events and perceptions, is sent to Action Selected, which will make the agent act on the environment [2].

  Action Selection has two different processes:

  – **Action-tendencies**, more instinct reactions like getting angry when you are attacked by a second agent, and,

  – **Coping Behavior**, reactions that have some reasoning behind them, resulting from the internal goals of the own agent, for example asking the second agent to stop.

  Coping Behavior can be divided also into two types of coping: Problem-Focused coping, involving planning and acting to reach goals, and emotion-focused coping, in which the agent's interpretation of the environment is changed. If the agent doesn't stop, the damaged agent can feel despair by failing to complete his goal, and lower this goal's importance, effectively reducing the despair caused by the failure [25].

2.2.3 Fuzzy Logic Adaptive Model of Emotions

Fuzzy Logic Adaptive Model of Emotions (FLAME) is a system, based both on a model of emotions, and, on the theory of cognitive appraisal. When a new event is occurring, FLAME estimates an emotional value of that event in relation to the agent's goals [23]. This process takes into account the emotional value of each event.

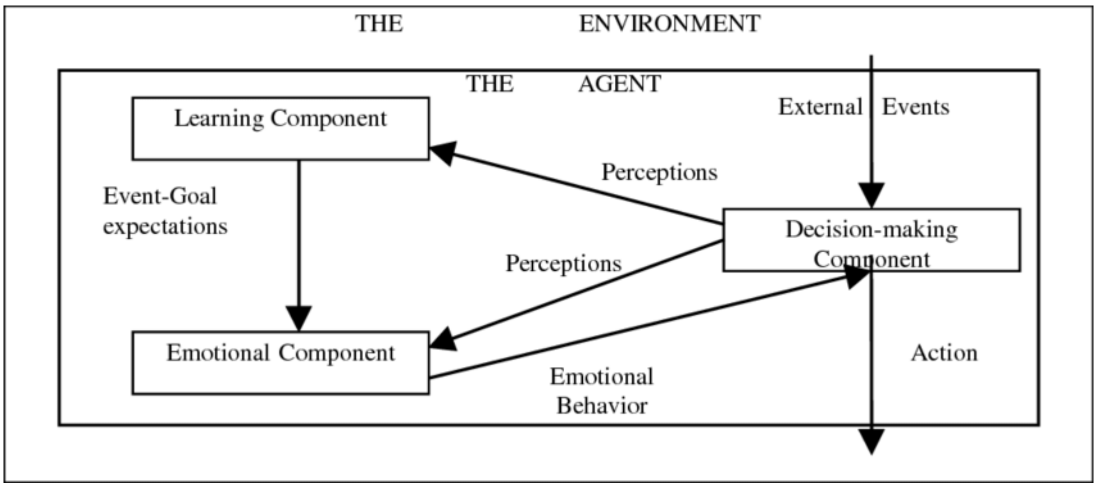


Figure 2.3: FLAME Diagram [3].

This architecture consists of three main components:

1. Decision-making Component,
2. Learning Component, and,
3. Emotional Component.

The agent perceives events from the environment, and they will influence these external information to the Learning Component and to the Emotional Component.

The learning component determines which goals are affected by the event and the significance that the event holds on these goals. The emotional component will compute and measure the perceptions, with the outcomes of the learning component, including expectations to produce an emotional behavior. This behavior is returned back to the decision-making component to choose an action.

The decision made is influenced by the current situation on the environment, the agent's mood, the emotional states, and the emotional behavior; then an action is triggered. Citing the example from [3], "every time you take the food dish away from your pet, you hit it to prevent it from jumping on you (decision made by the pet to try to recover the food dish while it's hungry). Taking the food dish away produces anger because the pet will be feeling both reproach and distress. The pet will feel reproach because, by nature, it disapproves of the owner's action (taking the food dish away), and it will be distressed because the event is unpleasant (getting no food to eat). Additionally, since the owner hits the pet whenever he/she takes the dish away, fear will be produced as a consequence. Thus, taking the dish away will produce both anger and fear."

FLAME was also utilized as a driver in virtual simulations for a virtual pet, which could perform actions such as barking or sniffing, jumping and playing, etc. The pet could also express emotions. The pet's behavior, and their appearance and credibility, were improved with significance by these emotions created with FLAME [23].

2.2.4 CIF - Comme il Faut

According to its authors, Comme il Faut (CiF) "is an artificial intelligence system and authoring strategy for creating game-based interactive stories about relationships and social interactions between characters" [26].

CiF enables a new class of interactive stories outside the purview of exploration and combat, and provides playability by focusing on social interaction; the player's actions have a deep impact

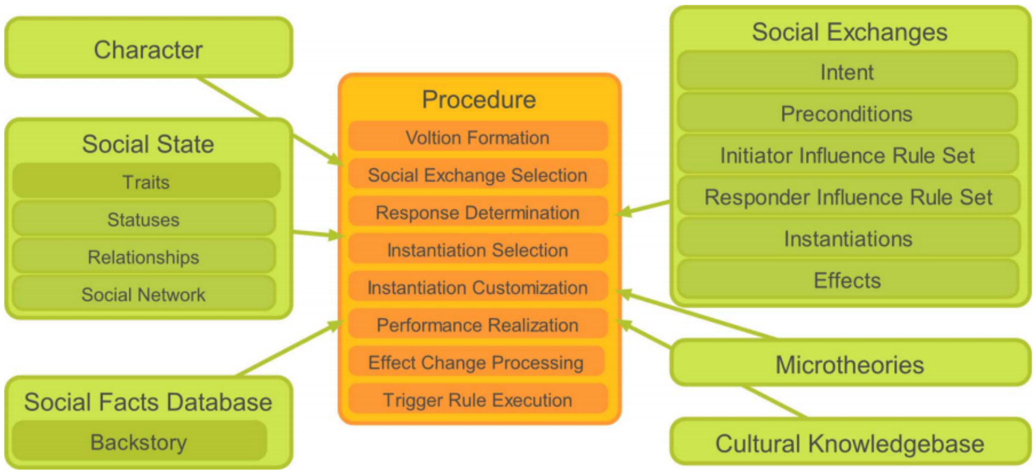


Figure 2.4: CiF Architecture [4]

on the social world and can greatly influence the social future of characters in a CiF-powered game. Social relationships are critical to many stories and are involved in nearly every story, which makes CiF an important step toward making stories more interactive [4].

Instead of compressing all domain knowledge in nodes or states, like many AI techniques do, such as Behaviour Trees and Hierarchical Task Networks, CiF chooses characters' behaviors based on rules in a large rule base that depict normal social behavior in a particular story world [27].

Unlike traditional game approaches to NPCs, CiF does not aim to create a branching storyline, but rather to establish the logic of a social world, and create characters and corresponding goals for each scenario. Since CiF is driven by a social simulation system, goals may be met in unexpected ways, but according its authors, they will be consistent with the defined social logic.

CiF operates by looping through a set of the process:

- Desire formation determines a character's desire, or volition to perform a social interaction (Social Move) with another character.

- A Social Game, or Social Move is selected for execution. A Social Move has an intent and three participants: the initiator, the responder, and optionally a third party participant.

- Determine how the responder reacts to the interaction The responder can accept or reject the social move

- When it's known how the Social Move is going to go, CiF needs to instantiate the performance, by executing a set of dialogue acts and corresponding animations.

- The social state effect associated with the chosen Social Move is applied to the social state and stored in in the system's memory.
- Finally, a set of "trigger rules" are executed to determine if additional social effects need to occur. According to McCoy et al.: "Trigger rules account for social changes that result from multiple social games and other elements of the social state. For example, if a character will receive the status of cheating after starting a relationship with one character when they are already dating someone else." [14].

CiF has been used already on games with a high social focus such as Prom Week with great success [14]. Also, there was an attempt to build a text adventure game from the ground while taking into consideration the requirements of the social model, called "Mismanor" [15].

2.2.5 Ensemble

Ensemble, also known as Ensemble Engine (EE) [5], is inspired on CIF. It was introduced as a freely-available social physics' engine with focus on enhanced expressivity and accessibility. It includes more expressive rules, a new action structure, and an user-friendly authoring tool.

One of the primary keys of the Ensemble is to track the state of each character, and the "environment's rules" are stored in a Social Record. These rules, who also influence the

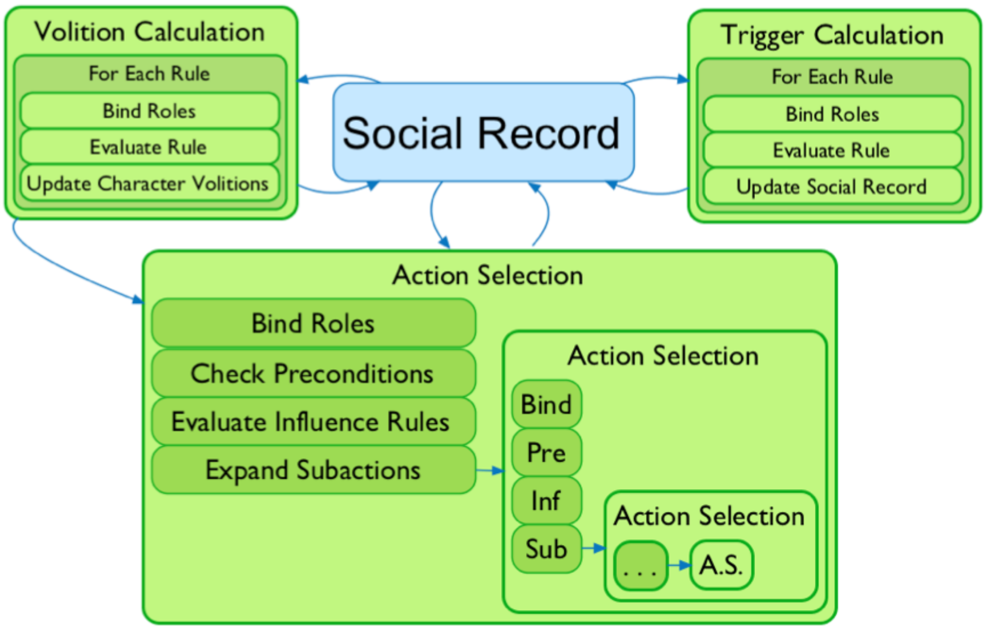


Figure 2.5: Three major processing elements of the Ensemble [5].

character's motivations, are divided into two different components.

The first component of rules is a sequence of predicates, where each predicate tests a single fact of the state of the environment. For example, in the next rule using three predicates:

1. A and B are not enemies
2. A and C are enemies
3. B and C are enemies

If all the predicates are true, then the second component of a rule will be executed. The second component can be composed of the following types of rules::

- Volition rules will either add to or detract a character's desire to perform certain types of actions.
- Trigger rules will directly change the environment's state by adding to or removing records from the Social Record.

While this process is very similar to the original CiF, there are some relevant modifications. For instance, regarding the binding process roles in rules, it is now possible to define any number of roles in a rule. From the previous example, in the rule using three predicates (1., 2. and 3.), all of these predicates involve two roles (A and B, A and C, B and C) but the role itself involves three roles (A, B and C). When on CIF authoring this event should be less convenient, meanwhile Ensemble can scale more than three roles easily (e.g. A, B, C, and D).

Once environment rules are used to determine each character's volitions and desires for social interactions, the next stage correspond to selecting actions that are appropriate to a particular social exchange. Previously, actions were connected to a specific intent, but on Ensemble they may be connected by one or more sub-actions, functioning as a hierarchy. Another improvement from CiF was the new system of actions that allows these actions to refer to more than two characters [5].

So, the rules on Ensemble are capable of describing more complex social situations involving multiple characters. Adding the actions that made it possible to refer to more than two characters, it makes more convenient to define the rules and to scale to more characters. Ensemble promises to be a strong architecture on social engagement of the characters on video games.

2.2.6 WASABI

The WASABI system is composed by three main components or processes. The emotional process produces a set of emotions based on relevant triggers, perceived from the environment.

The cognitive module, realized following the BDI (Belief-Desire-Intention) model, which creates signals of secondary emotions, and, allows a single action or a set to be performed. Finally, the third component connects emotion and action, and one of its functions is to generate speech actions taking into account the agent's emotions [23].

WASABI was used to model MAX, a virtual tour agent in a virtual museum environment. It has a reactive behavior, triggered when a new person enters inside the visual field of MAX. Also, it starts with a neutral emotional state until the player's greeting is processed by the agent. Depending on the player's dialogue (asking, insulting, and apologizing) MAX can react by getting annoyed, angry, surprised, or concentrated [6].

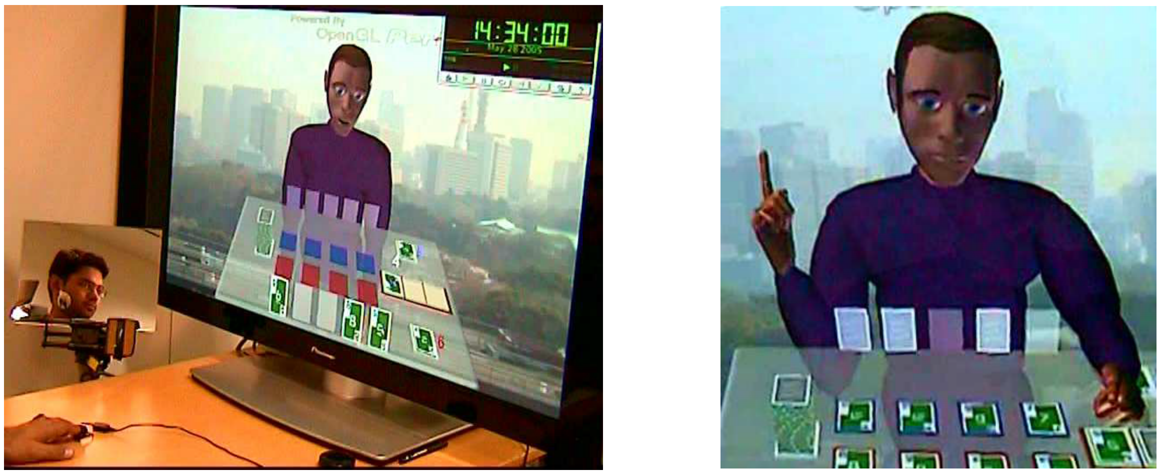


Figure 2.6: Interaction with MAX [6].

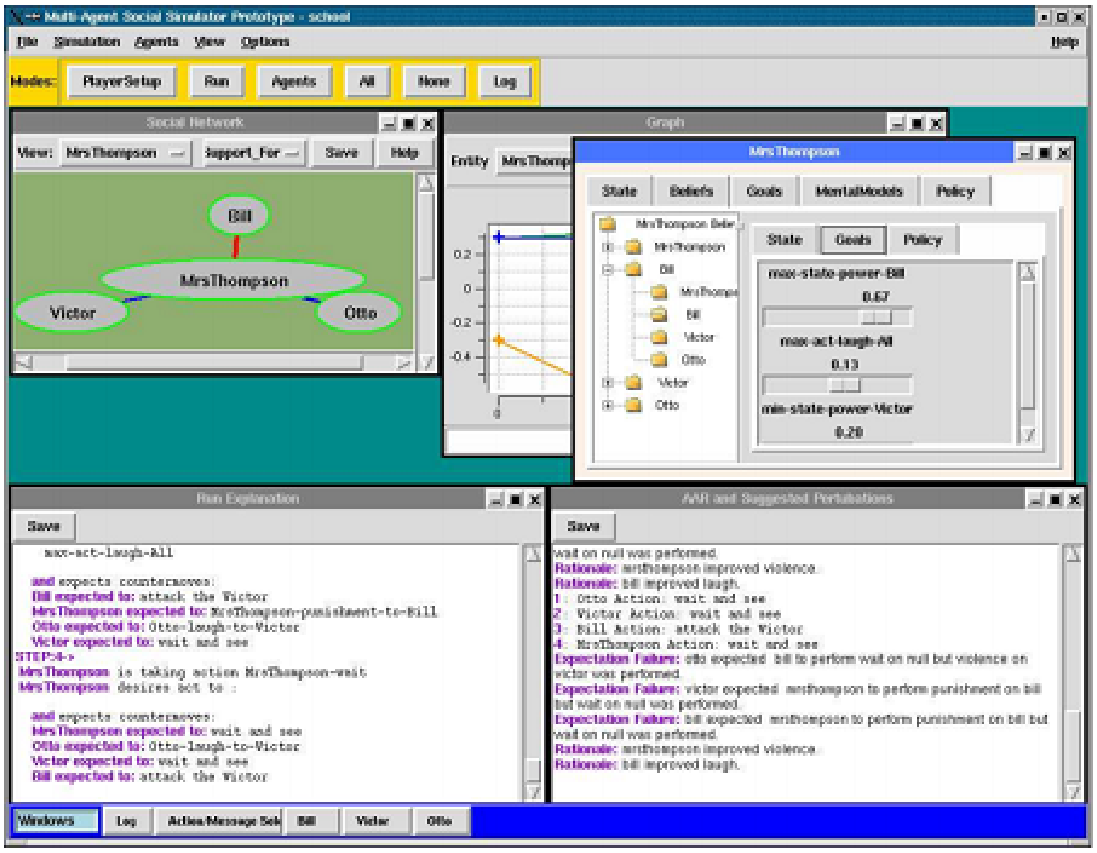
In the Figure 2.6, we can see the interactions between MAX and the player when playing a cards game, named Skip-Bo.

2.2.7 PsychSim

PsychSim is an agent-based system to simulate social interactions [16]. A unique aspect of the PsychSim design is that agents have fully specified models of others. These models are recursive and determine how the beliefs of an agent are affected by the agents around him [7].

The PsychSim Agent Models are composed by:

- State, several features representing its "true" state. This state consists of objective facts about the world, some of which may be hidden from the agent itself. In various PsychSim applications, it's possible to include state features like power, to represent the strength of an agent, economic resources, etc., to represent the various types of resources in a simulation.



```
  max-act-laugh-All

 and expects countermoves:
 Bill expected to: attack the Victor
 MrsThompson expected to: MrsThompson-punishment-to-Bill
 Otto expected to: Otto-laugh-to-Victor
 Victor expected to: wait and see
STEP:4->
MrsThompson is taking action MrsThompson-wait
MrsThompson desires act to :

 and expects countermoves:
 MrsThompson expected to: wait and see
 Otto expected to: Otto-laugh-to-Victor
 Victor expected to: wait and see
 Bill expected to: attack the Victor
```

```
wait on null was performed.
Rationale: mrsthompson improved violence.
Rationale: bill improved laugh.
1 : Otto Action: wait and see
2 : Victor Action: wait and see
3 : Bill Action: attack the Victor
4 : MrsThompson Action: wait and see
Expectation Failure: otto expected bill to perform wait on null but violence on
victor was performed.
Expectation Failure: victor expected mrsthompson to perform punishment on bill
but wait on null was performed.
Expectation Failure: bill expected mrsthompson to perform punishment on bill but
wait on null was performed.
Rationale: mrsthompson improved violence.
Rationale: bill improved laugh.
```

Figure 2.7: PsychSim Interface [7].

- Action, or a set of actions that can be chosen to perform in order to change the world. An action consists of an action type (e.g., punish), an agent performing the action (e.g. the actor), and possibly another agent who is the object of the action. For example, the action laugh(onlooker, victim) represents the laughter of the onlooker directed to the victim [7].

- Goals, in PsychSim's decision-theoretic framework, are represented as a reward function that maps the current state of the world into a real-valued evaluation of benefit for the agent. For example, a goal of Maximize/minimize action(actor, object) that corresponds to a positive/negative reward proportional to the number of matching actions performed, like maximizing their own economic resources, or minimizing the number of times any student teases any other.

- Beliefs, the simulation agents have only a subjective view of the world, based on their perceptions, where they form beliefs [7].

PsychSim allows a user to quickly create a social scenario where a diverse set of entities, individuals or groups, interact and communicate. Each entity has its own preferences, relationships, such as friendship and authority, with the other entities. The simulation tool generates the behavior for these entities and provides explanations of the result in terms of each entity's goals and beliefs [7, 16].

2.2.8 KARO

KARO is a merge of modal and dynamic logic, and accessory operators of motivation. The emotions are modelled using a formal logic-based language representation. A formalization of several operators that represent processes of knowledge and reasoning (Knowledge, beliefs, actions, abilities, desires) is provided by KARO.

The system uses four emotions that are not actually connected to external objects or actors. These emotions are related with specific attitudes, and they are triggered depending on the circumstances, as described in [23]:

- "happiness, triggered by achieving goals,
- sadness, triggered by failing goals,
- anger, triggered by being frustrated, and
- fear, triggered by goal conflicts or danger".

The model KARO was evaluated by implementing it on the iCAT platform [28] - a cat-shaped robot able to display facial expressions, and observing participant's interactions with the robot.

2.2.9 MAMID

MAMID is an agent archictecture based on the universal model of the (Belief-Desire-Intention) BDI-agent [29]. Additionally, the MAMID system has an explicit attention mechanism (with both processes filtration and selection), and expectations as supplementary phases. MAMID also supports. long-term memory storage of beliefs and rules.

Emotions in MAMID are generated based on internal interpretations, desires, priorities, and personality. This system implements the appraisal theory of emotion, and it uses cognitive and motivational elements to create emotions. But there are just four specific emotions, such as, fear, anger, sadness, and joy.

Thus, MAMID is a decision-making model for an agent in a certain experimental scenario. The agent's goal is to stay calm in several situations, and the different emotions, that it has, are activated, being influenced by the agent's personality and the current situation. [23].

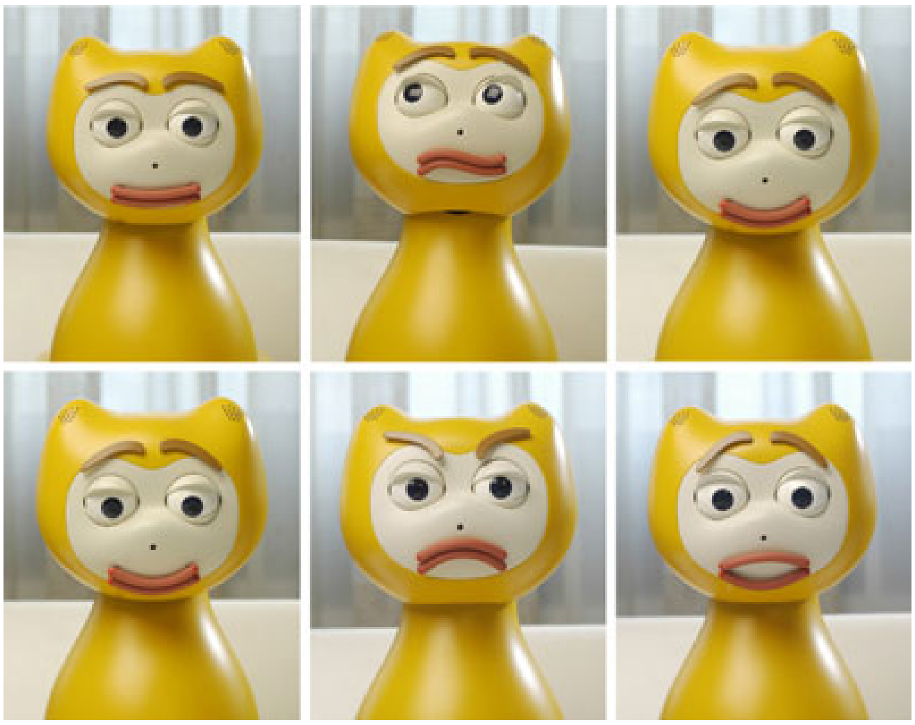


Figure 2.8: iCat Facial Expressions [8]

2.3 Social Focused Games

In this section, we will take a look about some games where the NPCs' behavior is not simply scripted but have social concerns and perform social actions.

2.3.1 CiF Application on Commercial Games

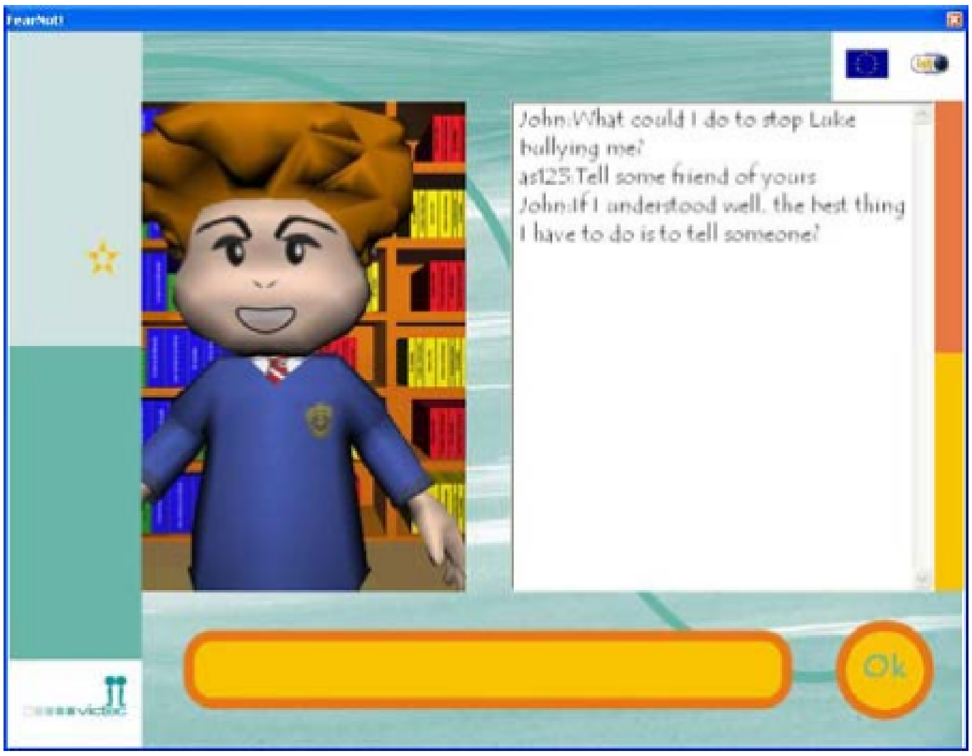
The CiF architecture was already used before on some popular games for social AI. For example, CiF-CK [16] which was implemented on The Elder Scrolls V: Skyrim [30], and more recently CiF-Ex [25], implemented on Conan Exiles [31]. For the sake of brevity, only the most recent CiF application will be discussed.

CiF in the Conan Exiles video game, works in the same way as CiF itself and by checking if there are new entries to the Social Exchange Memory of the character. Also, the process to the Trigger Rules is equivalent. The character handles the new memory by checking the agents names, Social Exchange name and Intention. Social Network Beliefs are changed during this process [25].

The CiF-Ex architecture features five main components: Social Exchanges, Characters, Social State, Trigger Rules and Microtheories. The initiating character starts a Social Exchange with the role of the initiator, setting the second character with the role of the receiver. The Social Exchange's Influence Rules use the characters' attributes (like gender, status, traits and memory), the Social State and Microtheories to compute the volitions, quantifiable representations of the NPCs' desire to perform a Social Exchange [25].

2.3.2 FearNot!

FearNot! is a PC application, based on FAtiMA, that sustains the removal of violence from schools. It works as a simulation, where you sympathize with a victim of bullying. To reach this sensitive goal, the application should have credibility both in terms of behavior and emotions. These emotions are generated based on cognitive appraisal of the current experience of the character [23].



John:What could I do to stop Luke bullying me?
as123:Tell some friend of yours
John:If I understood well, the best thing I have to do is to tell someone?

Figure 2.9: FearNot! gameplay. User interacting with a Character [9].

The virtual stories experienced in FearNot! let the children witness (from a third-person perspective) a series of bullying situations towards a character. However, instead of just watching, the child discusses the problems with the victim and proposes coping strategies [9].

Children's suggestions to the main character, will change his internal importance of goals. For instance, if the child suggests the victim to fight back, he will increase the importance of fighting back, making it generate stronger positive emotions, which will override the fear generated by the prospect of fighting back. So, next time the victim is confronted by the bully, he will have the courage to fight back.

2.3.3 ORIENT

The Overcoming Refugee Integration with Empathic Novel Technology (ORIENT) agent mind is built upon FAtiMA [2] architecture applied in FearNot!v2.0. FAtiMA was developed with the goal of creating believable synthetic characters, where emotions and personality play a major role in the character's decision-making processes and are visible through the character's actions [10, 32].

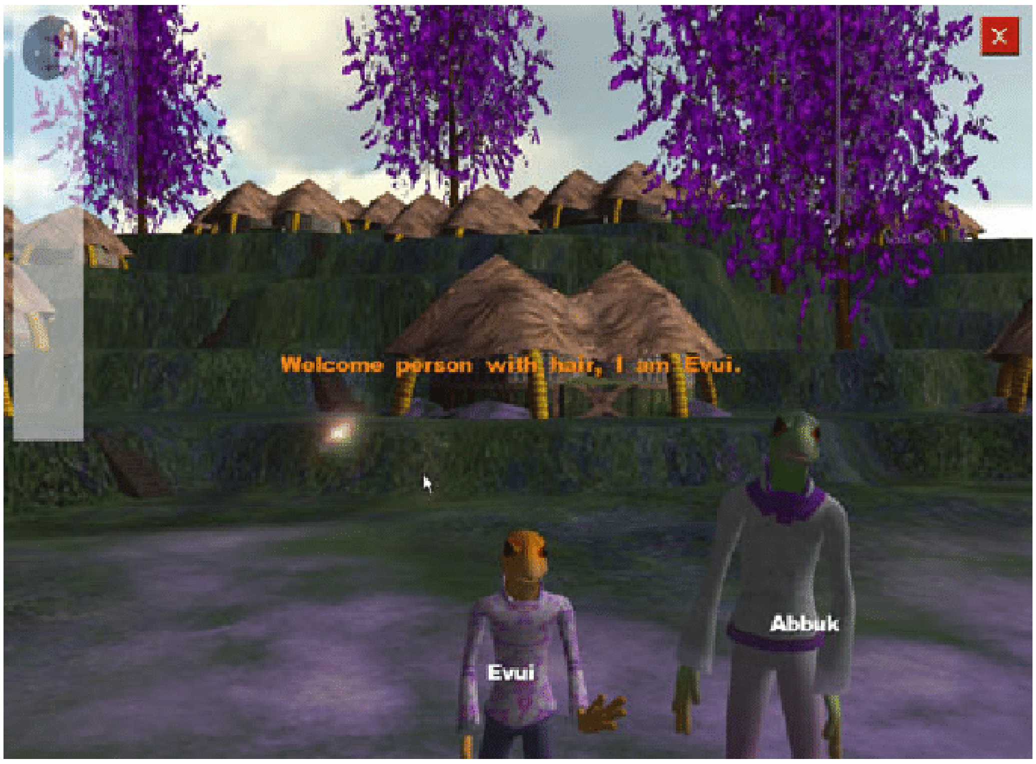


Figure 2.10: Sprytes in ORIENT [10].

The ORIENT scenarios show how imaginary people (the Sprytes) struggle to reconcile their own interests, attitudes and feelings with another culture [33].

By assigning different weights and decay factors for different drives to different ORIENT agents, characters with different personalities can be produced. For example, if character A is a friendly character, affiliation would be an important factor in its social relations, say weight 0.7 while a hostile character B would have a low importance for affiliation, say weight 0.3. What this means is that character A will have a higher tendency to show friendly behavior compare to character B, although responses may differ depending on circumstances and other factors such as like-dislike relationship.

The characters on ORIENT are making predictions constantly about the probability of success of their goals, determined by the ratio of success and number of tries. Then these predictions are compared with the current output (success or failure). The difference between the prediction and the output is the observed error, and using the current and previous observed errors, it will be possible to determine the uncertainty in the prediction of goal success. Getting certainty, it is achieved by exploration of new strategies and actions, corresponds to creating a more accurate model of the environment. As such, goals that are relatively unexplored or that present higher error estimation (higher uncertainty) present more potential for improvement.

Also, the character learns by trying out different actions from a pre-specified set of actions and remembering which actions helped it to tackle a situation best. For instance, in the previous example, character B will remember that showing anger towards the user will not help it to restore its integrity. This information is stored in the memory and is used to determine the success probability of actions/goals in satisfying a specific drive in the future. Since certainty depends on the amount of known information relating to a goal, the more an agent encounters the same type of situation, the higher its certainty is regarding the situation [10].

2.3.4 Façade

Façade is an open-ended dramatic scene between the player and two game-controlled characters. The drama takes place in the apartment, a small simulated virtual world, of a couple Trip and Grace, where the player is rewarded for being proactive [16]. Grace and Trip can use objects, move anywhere continuously, speak and gesture anytime. After the simulation begins, the couple will quickly start arguing. The characters respond to a variety of open dialogue that the user can input, including questions and insults, using the keyboard. Through dialogue, gesture and action, the player has continuous effects on her affinity with Grace and Trip, the current level of tension and the specific information revealed about their marriage.

The Façade architecture consists of characters written in the reactive-planning language ABL, A Behavior Language, who is a programming language explicitly designed to support programming idioms for the creation of reactive, believable agents [33].

Façade still had its problems because it uses open dialogue in order to communicate with the characters, and it's not always successful in translating the intentions of the player. Also, even with lots of hard-coded template rules, the player will be able to input text that the system

Figure 2.11: Façade Screenshot [11]

cannot understand and when it occurs the best the system can do is fail gracefully [34].

2.3.5 The Sims Series

The Sims series employs a social simulation model, players control their own Sims' activities and relationships in a manner similar to real life. The gameplay is open-ended and does not have a defined goal. Challenges occur randomly based on aspects of each Sim's lifestyle, such as relationships, skills and job.

The approach of The Sims is largely focused about what is happening on the current instant, even the physical, while extremely dynamic. There is no evident path in the approach of the game The Sims 3 toward the kinds of significant character histories that are highly considered to be crucial for richly realized characters.

The commercial successful video game The Sims is an example of a game that has a very dynamic social environment [16]. The game's characters, named Sims, have traits and desires that influence on their social moves. These social moves are explicitly modeled after social standards, which are described as hierarchies of expectation clusters. These moves have both a regulating and constitutive impact on character conduct: they help define how an agent must not behave, as well as allows new behaviors and intentions. The practice can discourage

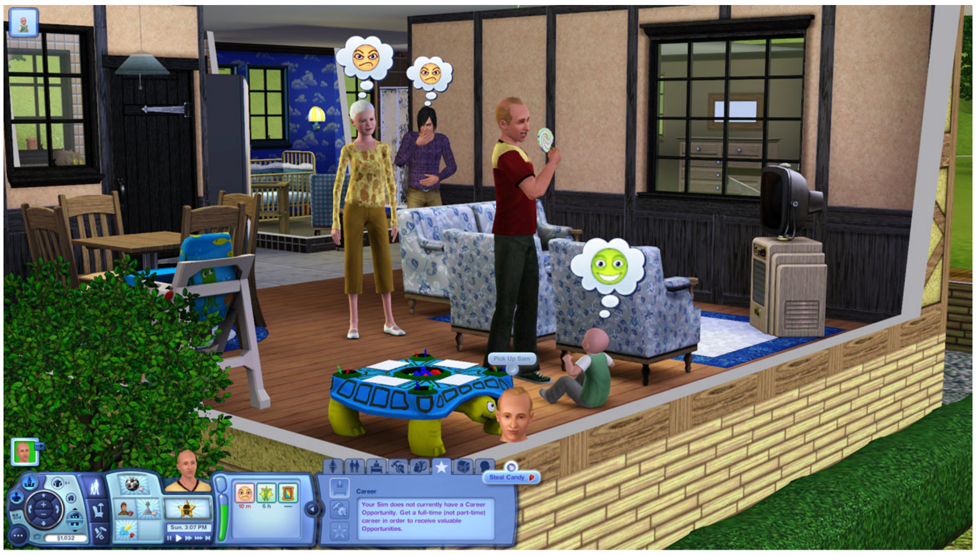


Figure 2.12: Sims Screenshot [12]

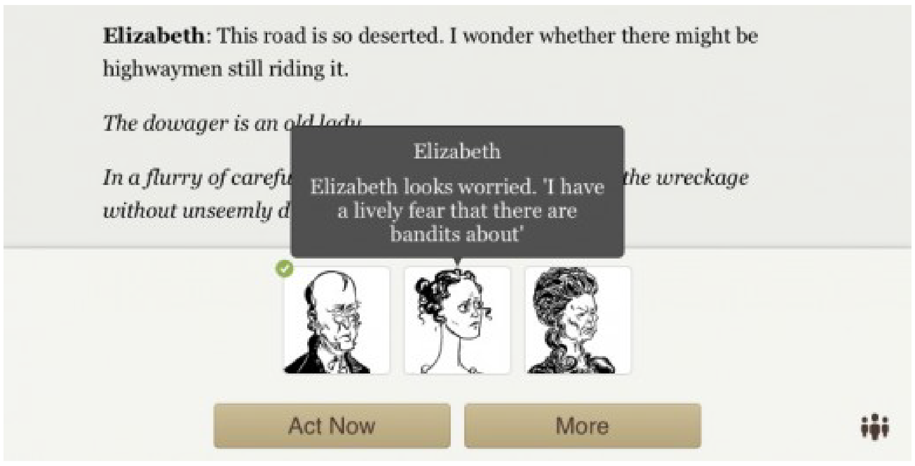
certain actions and encourage others. It can detect when norms have been violated and respond accordingly, by changing state and sending new requests. For example, when it's bedtime, the parents tell the child to go to sleep. If he doesn't sleep and leaves the bedroom, the social state of his parents will change, to have the goal to remind the child to go back to bed, and taking him back if he refuses.

The Sims distinguishes between primary and secondary emotions. As example, given that Sims considers Anger as a primary emotion and Happiness as secondary emotion, in the case where the sim has +4 to happy but only +2 to angry, they will stay positive despite having an angry mood. If some event occurs and the balance shifts to +2 happy and +2 angry, the sim will become angry because happy is no more dominant (dominant emotion which determines what a Sim is feeling) and angry is a primary emotion. If two positive emotions are tied, the first that came will determine the Sim's Emotional State [35].

A Sim can participate in many practices at once, allowing the the characters to experience conflicting goals and intentions [17].

2.3.6 Versu

Versu is an innovative platform for interactive stories. It's a text-based interactive simulation drama [13]. It uses autonomous agents and players can play the same story with from different perspectives, and attach different characters to the various roles. Every person in one Versu story is formed with AI that provides them unique personality traits. They will remember how


Elizabeth: This road is so deserted. I wonder whether there might be highwaymen still riding it.

The dowager is an old lady

In a flurry of carefu *the wreckage without unseemly d*

Elizabeth

Elizabeth looks worried. 'I have a lively fear that there are bandits about'

Figure 2.13: Versu game interface [13].

the player has treated them. It is possible, also, to have them as a friend, a lover, a mentor, or, an enemy. They perceive what you say, but they are able to also respond to gestures and expressions.

The AI engine designed by Richard Evans, the lead AI designer for Sims 3, which allows each character in a story (and in some cases a drama manager AI) to act autonomously or be played by a human player [36].

The used architecture in Versu, uses social practices which are implemented as reactive joint plans, providing opportunities to the agents. These social practices don't control the agents directly but supply suggestions, and, it is the agent who decides what to do [13].

A social practice describes a type of ongoing social situation. Some of them (e.g. conversation, a meal, a game) only exists or a short time, while others (e.g. a family, the moral community) can last much longer. A social practice provides the agent with a set of suggested actions, but it is up to the agent himself to decide which action to perform, using utility-based reactive action selection. Agents score each available action and then execute the highest scoring action. For example, the characters can play a cards game like Whist. When considering the cards that the whist-player can play, the decision-maker evaluates the features of a move (whether it counts as winning-the-trick, whether it counts as throwing-away-a-card, trumping, getting rid of a suit, etc.). These conditional effects determine the score of playing a card. Using such simple appraisals, and giving to the characters suitably-weighted desires to execute actions which satisfy these appraisals, is all that is needed for the characters to play a game automatically [37].

2.3.7 Prom Week

Prom Week [14] is a social simulation game about the interpersonal lives of a group of high school students in the week leading up to their prom, based in CiF social architecture, which was already discussed before.

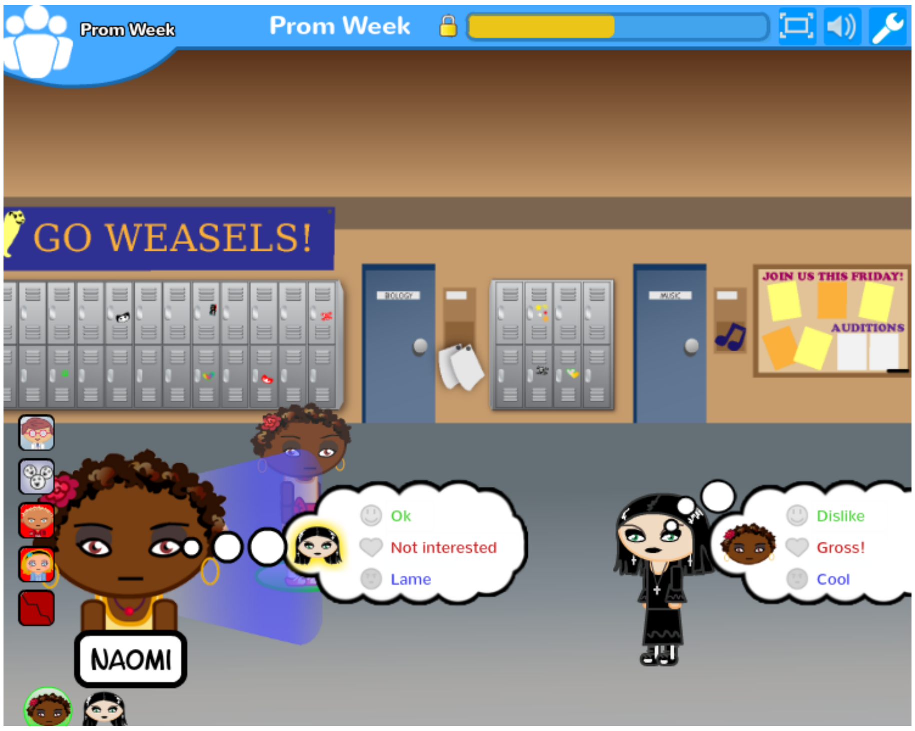
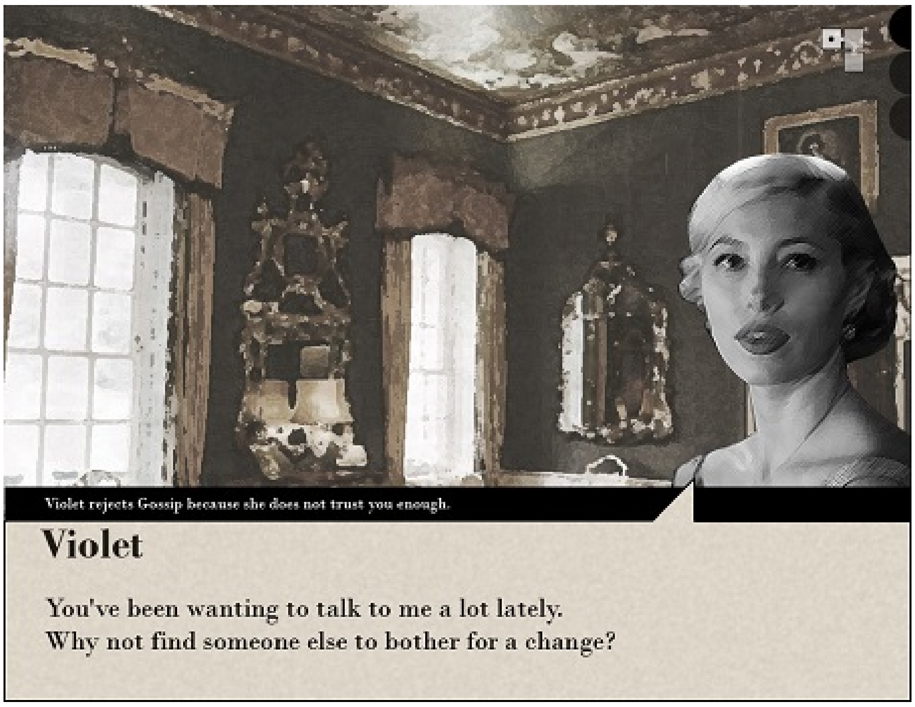


Figure 2.14: Prom Week [14].

In Prom Week the player doesn't have an avatar, instead he controls all characters in the game, in order to solve level goals such as making the stereotypical class nerd, Zack, date a popular girl within a limited number of turns. Prom Week uses the social physics of CiF for maintaining a highly dynamic story space, while each level goal has a high number of possible solutions that maintain character believability. CiF also provides and manages which social exchanges are available and how each one changes the social state. [14]

2.3.8 Mismanor

Mismanor is a role-playing game in which the core mechanic is social interaction instead of combat, with a focus on supporting player-driven story, and, it is also a modified version of CiF architecture [15]. The game is a historical fantasy, set at a manor in the countryside. There are six characters the player may interact with including a Colonel, his beloved daughter, Violet,


Violet rejects Gossip because she does not trust you enough.

Violet

You've been wanting to talk to me a lot lately.
Why not find someone else to bother for a change?

Figure 2.15: Mismanor [15]

the stable-boy, James. As the player character interacts with the family, it is gradually revealed that some game characters are members of a cult, and the player character was invited to the dinner party for not entirely innocent reasons. Interaction between the player character and game characters takes place through dialogue exchanges. This modified version of CiF, used on Mismanor, supports characters in the creation of desires to perform actions related to acquiring and sharing information [15].

2.4 Discussion

During this chapter we discussed different *Social Emotional Models* like FAtiMA, CIF, PsychSim, EMA, ParleE, MAMID, Wasabi, and some *Examples of Social Emotional Models Application*, such as Orient, who is a FAtiMA's application, and Wasabi.

When choosing one social architecture to implement on this work, it is important to match the purposed goals with the system functionalities.

The focus of this work is the creation of social NPC's for Bannerlords, to make them more interesting and believable, and thus improve user's experience. In the author's opinion, the architectures that were more appropriate to be used for this effect are FAtiMA and CiF because they have been used already in different contexts and served as inspiration for other models and applications.

While FAtiMA is more focused on the generation of emotions, and how those emotions may affect the behavior of characters, CiF has an explicit representation of social relations between NPCs and how they can influence behavior.

Due to this, and to the fact that we want to focus and improve the social interactions and the social relations between the NPCs themselves and between the players, we are choosing the CIF architecture to implement an AI Social Architecture on Bannerlords.

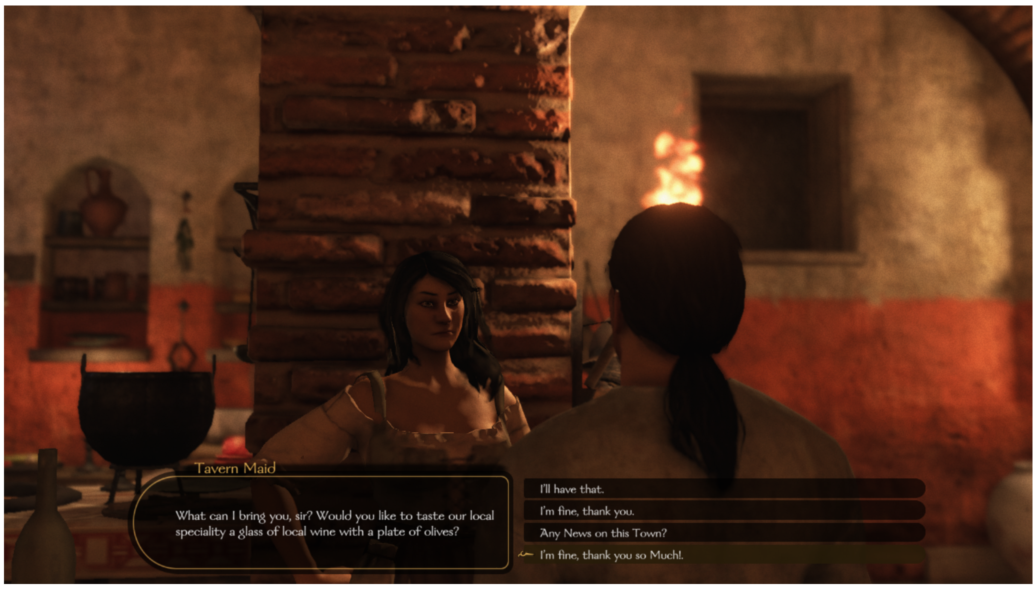
3 Mount&Blade II: Bannerlord

Contents

In this chapter, we provide a brief description of the game and study its architecture.

3.1 General Description

Mount&Blade II: Bannerlord is a strategy action role-playing game being developed by TaleWorlds Entertainment, at the moment. Bannerlord is the eagerly awaited sequel to the acclaimed medieval combat simulator and role-playing game Mount&Blade: Warband [38].


Tavern Maid

What can I bring you, sir? Would you like to taste our local speciality a glass of local wine with a plate of olives?

I'll have that.
I'm fine, thank you.
Any News on this Town?
I'm fine, thank you so Much!.

Figure 3.1: Dialogue Screenshot.

Figure 3.1 depicts an example of Bannerlord's dialogue system, and an interaction between the player's character and the tavern maid.

The NPCs that interact the most with the player are the quest givers, companions and nobles, and most of that interaction is instrumental rather than social, as discussed before. For instance, if the player completes a quest for a quest giver, their relation will increase a few points and that will allow the Player to have additional troops for recruitment. Also, the companions will have only the behavior to follow the Player through the game. There is no real social interaction with characters, and the Player doesn't have the freedom to socialize with them. For last, Lords will rule different settlements and lead wars against each other. In order to join a faction, the Player have to swear loyalty to the ruler of said faction, and when create its own kingdom, the Player can attempt to convince Lords and their Clans to join the kingdom.

Bannerlord offers an open world experience - a game mechanic of using a virtual world where the player can explore and approach objectives freely, as opposed to a world with more linear and structured gameplay. Although the player can complete quests, there is no overarching storyline present. The player is able to join one of the factions, fight as a mercenary, assume the role of an outlaw, or remain neutral.

This game has a community modding and the game supports also the developers, giving tools and being available to test and debug the mods made by the community. Also, on launcher, it is possible to choose about which mods the player wants to play with before the game start itself.

3.2 Bannerlord Architecture

Unfortunately, there is no existing documentation publicly available about the source code, or even the game's architecture. In order to understand how to implement and integrate our mod, a reverse engineering process was used to extract the code from the binary files. In this section, we provide an overview of the general structure of Bannerlord's architecture. A more detailed discussion of this process will be presented in Chapter 5.

The Bannerlord's architecture, is divided into a set of sub-systems, which are incorporated into the game as Dynamic Linked Libraries.

- **Core**: contains and defines basic data structures used throughout the game - characters, items, missions, weapons and quests data, etc. It also prepares, declares and defines default codes, definitions, and constants to be used during the game;

- **Sandbox**: this sub-system is used to control the behavior of the game that is not interfering with the main story, like common characters on villages, towns, etc.;

- **CampaignSystem**: where you will find everything and the related code about the main story and the main quests. The Campaign System has some influence on Sandbox. For instance, when a vulgar NPC talks about the previous quest or about the current one, it needs to consult the CampaignSystem to determine this information.

- **MountAndBlade**: this subsystem is majority responsible for the implementation of the multiplayer system, allowing the player to play online against other players. It contains also some structures used throughout the game - agents, behaviors, tactics, formations, etc.

In the Figure 3.2 it is possible to have an overview of how the sub-systems are connected and how the game works internally.

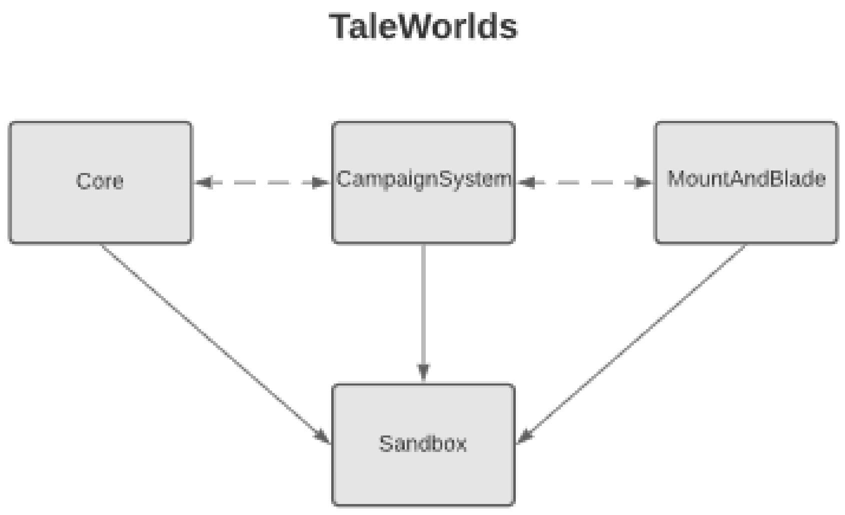


Figure 3.2: TaleWorlds subsystems.

The agents' AI that controls the NPCs behavior is declared on MountAndBlade subsystem. When the NPCs are created, they are spawned on their coded locations, and before they start moving the dialogues are loaded and stored. These dialogues will only be displayed when a specific event occurs, in this case it's when the player interact with the NPC.

Although when loading the game all the NPCs and their dialogues are created, they are only spawned when the player enters into a town or a village. Specific NPCs for each location are spawned, and their dialogues are displayed when the player interacts with one of them. Each different type of NPC, defined by the game, has its own set of specific dialogues (tavern maid, arena master, townsman, etc.).

The NPCs, e.g. the common villagers, dialogues are created on Sandbox subsystem (code located on SandBox.Source.Towns inside Sandbox.dll). It can be found also the methods used to create and add characters to their default locations. Meanwhile, their information filled with the current *CampaignSystem* or, by default, with the *Core*'s information.

On Sandbox component, it is initialized also some of the character's behaviors, being defined on *AgentGroupBehavior* who will influence the character's behavior. It uses attributes from the other subsystems.

Some of character's behaviors are, for instance:

- *FollowAgentBehavior* used to follow a character,

- *WalkingBehavior*, character moves randomly on the current map,

- *EscortAgentBehavior*, to escort a specific character until a defined position,
- *FightBehavior*, to fight against the opponents on arena or on battlefield, and,
- *PatrollingGuardBehavior*, used to patrol posture and move into defined positions, specific for the guards by default.

4

CIF-BL

Contents

In this chapter, we present and discuss the adaptation and implementation of the CIF model into Bannerlords, which we named CIF-BL.

4.1 From CiF to Bannerlord: CIF-BL

The original CIF model, and other agent architectures, were designed to autonomously generate behaviour independently of how many agents there are in a particular environment. However, in the case of Bannerlord, an AAA open-world game with many NPCs, the frequency of interactions and efficiency of using computational resources becomes of paramount importance. For instance, if the player is in a tavern or village with tens of NPCs, and they all start talking at the same time, it will be very hard for the player to grasp what is going on. According to Ian Millington [39], due to the perception Window effect where the player only pays limited attention to NPCs, the AI level (or complexity) of a character should consider the level of attention it will get from the player.

Due to this reason, we felt the need to not only manage the Social Exchanges individually, but to create a manager that is able to manage all interactions in a centralized way.

The CiF architecture decides also, from the NPCs surrounding the player, which one will perform a social interaction, until a defined distance. It also notifies the other NPCs of any social state change, allowing them to update accordingly.

As displayed in Figure 4.1, inspired in the CIF model, the CIF-BL architecture has five essential components:

- a Character which represents a structure that encapsulates the social characteristics of an agent (NPC);
- a Social State that represents the character's current state of their social relations;
- a Social Exchange (SE) that represents possible social interactions between characters to achieve their goals;
- Trigger Rules which trigger on specific events (as a reaction to the changes of the environment).
- CIF-BL Manager, a controller component responsible for managing the activation and execution of SEs amongst NPCs.

The other components present on CiF architecture (such as Predicates and Time-Ordered rules) were not implemented on CiF-BL, because we did not considered them essential to the

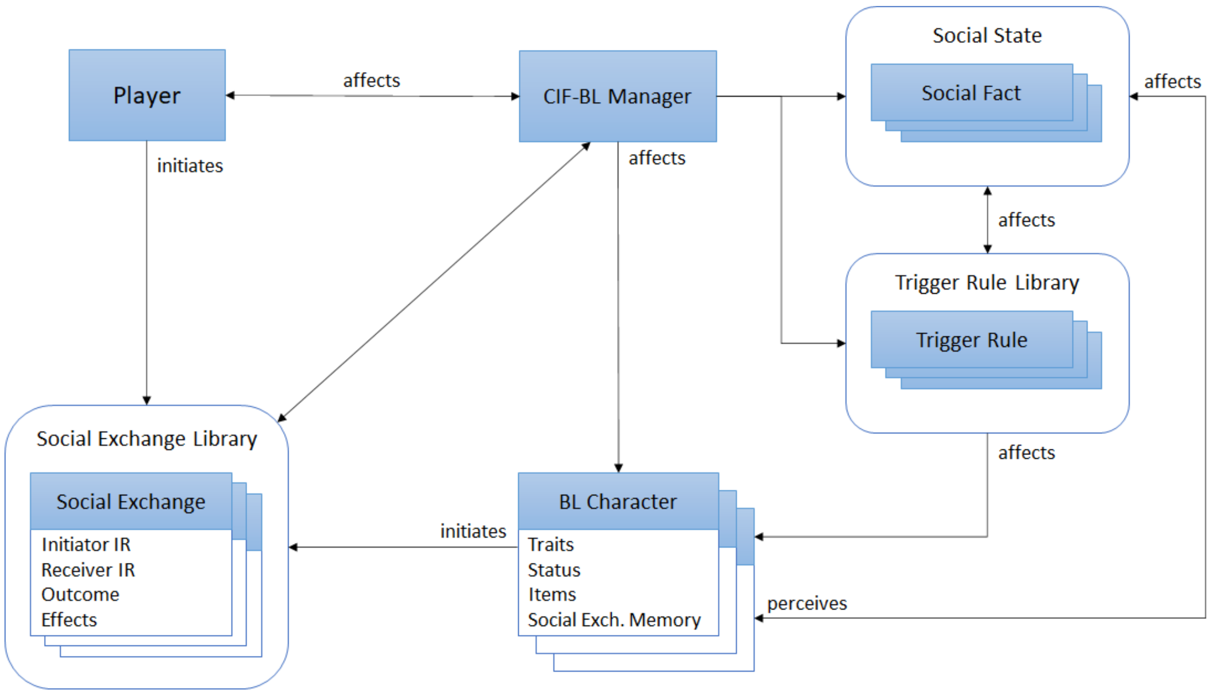


Figure 4.1: The CiF-BL Components.

model in an initial stage, and we wanted to maintain CIF-BL relatively simple. For instance, symbolic predicates would be useful if we would like to provide authors the capability to dynamically add influence rules to a character while the system is running. However, this is a not a crucial feature for our mod.

We start with a high-level overview of the model. The central component *CIF-BL Manager* is responsible for managing and controlling the decision formation process for NPCs, deciding which NPCs are active or not - according to their distance to the player, and updating information about the environment (e.g. social state).

In order to limit the amount of social interaction amongst NPCs - so that the player is not overburden with social interactions that he is not able to pay attention to - the *CIF-BL Manager* monitors the number of currently ongoing SEs, and does not allow it to go beyond a predefined maximum. Individual BL characters also have their own cooldown that prevents them from performing multiple SEs in a row.

If the maximum of ongoing SEs has not been reached, the *CIF-BL Manager* will activate the desire formation process for each character, testing which social exchanges might be desirable from them taking into account the current social state. The desire formation process will lead to the selection and execution of a Social Exchange with an *initiator* and *receiver*. When the Social Exchange ends, the *CIF-BL Manager* is notified and updates the Social State and the BL-Characters' memory (named as Social Exchange Memory on the Figure 4.1 that will record all the interactions and events taken from the environment that each specific BL-Character will see). Finally, it verifies if there are conditions available to activate a Trigger Rule on one BL-Character.

Given that the *Player* has its own initiative to start new Social Exchanges at any time, the central controller *CIF-BL Manager* needs to handle it separately. If the player starts a new conversation, the manager stops the desire formation process on *BL-Character* entities and focus on the Player's dialog, sending and receiving information about the decisions made.

We will now describe in slightly more detail some core processes and data structures of the proposed CIF-BL model, starting with the desire formation process.

4.2 Desire Formation

The original CiF system uses the Social Exchange's Rules (also known as Influence Rules) that based on the character's attributes (gender, traits, statuses, and memory) calculate the volition (on the *calculateInitiatorVolition* function present on algorithm 4.1). Volition represents a quantifiable representation of the NPC's desire to carry out a SE.

Algorithm 4.1: Desire Formation - CiF-BL Pseudocode

```
availableSE ← {}
if OnGoingSEs < MaximumSEs then
    for c in Characters that are active and ready and not busy do
        for $c_2$ in $Characters \setminus \{c\} \cup \{Player\}$ that are active and ready and not busy do
            for se in SocialExchanges do
                $se_i \leftarrow newSocialExchangeInstance(se)$
                $se_i.initiator \leftarrow c$
                $se_i.receiver \leftarrow$ $c_2$
                $sei.initiatorVolition \leftarrow calculateInitiatorVolition()$
                if $se_i.initiatorVolition > 0$ then
                    availableSE.add($se_i$)
    if $availableSE \neq \emptyset$ then
        //maxVolition returns a set, there might be more than one SE with equal maximum
        topSE ← $maxVolition(availableSE)$
        nextSE ← $random(topSE)$
        getNPC(nextSE.initiator).startSE(nextSE)
        OnGoingSE ←OnGoingSE + 1
```

On the algorithm 4.1 is possible to see how the CIF-BL Manager controls the Desire

Formation of BL-Characters. The algorithm starts by resetting the auxiliary variables and checks if the number of ongoing SEs is less than the defined maximum or not. If returns true, so it begins to calculate all the possibilities with the BL-Characters that are active, already not interacting with someone else, and ready to perform a new SE with another BL-Character. Each BL-Character calculates how much volition each SE has to perform with the other surrounding characters.

The Social Exchange's collection it's defined on the CIF-BL Manager's initialization, and it contains all the SEs available to be performed on the game.

After calculating all the possibilities, the availableSE's collection contains the Social Exchanges that have the biggest volition (possible to get with *maxVolition(availableSE)*), and a random SE will be picked from this set to start executing (by running *random(topSE)*. The initiator defined in the selected SE will be the NPC which will execute the SE (as the initiator).

Being an additional challenge in the implementation of CIF-BL was the need to consider two different dialog systems for the Social Exchanges: for the interactions between two NPCs and between an NPC and the player. One of our goals was to design the system in a way that the player and NPCs are treated similarly. This means that NPCs needs to respond to a player's social interactions in a similar way how it would respond to an NPC. The NPCs' must be able to consistently react to the player when he is aggressive or friendly, in the same way they would react to other NPCs. They will be able also to dialogue with other characters, to have a good and not good friendships with them, and also start dating and breaking up with each other.

4.3 BL-Character

A **BL-Character** is a structure that holds each NPC's social characteristics. The data described in Table 4.1 allows the architecture to add social behavior to the NPCs.

We improve the Characters created in Bannerlord by adding additional behavior. While characters in game already have some elements required to implement this architecture, we need to further expand upon those by adding a number of CiF based properties.

- **Name**: Character Identifier, used to know which characters are interacting. Uncommon characters, known as Hero, normally are unique characters and the ones who have quests. Each one will have its own name.

  For the common Characters, there are multiple of them with the same name (multiple Townsman, Townswoman, Trader, Dancer, etc.) being impossible to identify which was the "Townsman" who the player was interacting with. So it was needed to create a helper

Term	Function
Name	Identifier
ID	Helper Identifier
Gender	Whether the NPC is male or female
Countdown	Stops the own character from starting SEs
Busy	Stops the other characters from starting SEs with this character
Culture	Origins Characteristics of the NPC
Traits	Personality Characteristics of the NPC
Social State	Social Characteristics of the NPC
Social Exchange Memory	Memories of all the SEs that they have perceived

Table 4.1: Characters in CiF-BL.

variable (an ID) for each character to double-check if the Townsman has the right ID and, respectively, if it is the character on the social exchange;

- **Countdown**: is used to verify if a character is resting. After performing a SE, the character goes into cooldown and isn't able to start SEs until the time expires.

  The time of the countdown is dependent on the character's traits (for example, a character with the trait Shy will have a higher than average countdown, and the opposite for the trait Brave); It is influenced, also, by the number of NPCs in the same location. For instance, outside the buildings there are more NPCs than inside them, so the countdown will be lowered, working similarly for the day/night cycle.

- **Busy**: used to check if other characters can start interactions with this character. If a character is set as the receiver for one currently ongoing SE, by another character or by the Player, they will be flagged as busy.

  The Player will be treated as a normal Character. It is able to start interacting with a character having an ongoing SE, but it will not affect the social exchange, just pausing it;

- **Culture**: defined by the game and re-used to influence the dialog lines on the social exchanges. For example, if both NPC A and NPC B will perform the same Social Exchange but having different Cultures, the sentences that each one can speak will never be the same. Each culture has it specific dialog sentences for each Social Exchange. They may affect also the character's volition depending on if the culture is friendly, unfriendly or neutral with other cultures;

- **Gender**: used to calculate Volitions, similar to Traits;

- **Traits**: used to give the characters a unique personality. Traits are permanent properties that impact social exchanges. A list of traits can be seen in Table A.4;
- **Social State**: contains the Status, Social Network Beliefs.
- **Social Exchange Memory**: holds the information about the SEs that happened before and that the character noticed. It works like a Character's Memory, and it's used to save the previous social exchanges performed with others (as initiator or receiver). It will also store the social exchanges performed between external characters if they are within the scope of their perception.

  Each one will have their own memories, they will memorize their previous social exchanges (as initiator and receiver) and a perception of what is going on near to them. We will discuss, later, how the memory will influence when calculating the volition to the next Social Exchange.

Characters should exhibit socially adequate behavior to fit the context they meet. Fitting the context is particular relevant for interactive agents that interact and are being observed by people [40].

4.4 Social Exchange

The primary knowledge representation in CIF is the social exchange, a collection of patterns of (primarily behaviors and dialogue) interactions where the exact performance and social outcome varies based on the personality-specific attributes of the characters involved (Traits) and the current social state (Status). It represents the social interactions that the NPCs and the player can have with each other.

The Social Exchanges implemented increase and decrease the relations on game, for instance, *Compliment* SE and *Jealous* SE.

The Table 4.2 briefly describes the internal structure of a Social Exchange, and a short description of the Social Exchanges on Table A.2.

As described on the Table 4.2, there are some essential components:

- **Intention** is used to generalize behavior when calculating Influence Rules. It is a keyword that represents the purpose behind the Social Exchange, for instance the *Compliment* SE has *Positive* as intention and the *Flirt* SE has *Romantic* intention.

  All the intentions with the respective Social Exchanges can be seen in Table 5.2.

- **Influence Rules** are general rules applied to the initiator and the receiver when calculating the volition for each SE [41]. An Influence Rule contains a collection of new conditions that influences the character's volition, depending on the traits and relationship. The *Initiator Influence Rules* are used to prioritize which Social Exchange the initiator wants to perform the most, and the *Receiver Influence Rules* are used to calculate how the receiver will react to the SE.

  When the character is able to start or react to one Social Exchange, the influence rules tests if it should or should not happen, by increasing or decreasing the volition. For example: *AskOut* SE should not happen if the NPCs are already dating, and, *Break* SE should not happen if the NPCs are not dating.

  If the characters are dating, they have more will to perform SEs with *Romantic Intentions* between both. If they are friends, they desire to perform SEs with *Positive Intention*.

- **Outcome** can be *Positive* or *Negative*, depending on the calculated receiver's volition, and it will dictate the own reaction. If the volition is less or equal to 0 the outcome is *Negative*, and, if it's higher than 0 the outcome is *Positive*.

  The outcome it's also used to select the Effects of the SE. They are the consequences of the SE and modify the participating characters' Status and their Social Network Beliefs. For each SE and possible Outcome, there are fixed values that are added to the *Social State* and *Character Status*.

When a Social Exchange is performed occurs, the process of adjusting the Status. The Outcome will apply a set of changes to the participants' Status. As such, there is one set for each Social Exchange and possible Outcome.

Term	**Function**
Intention	Keyword that defines the objective behind the SE
SocialExchange	Unique Identifier
Influence Rules	Collection of rules that returns the character's volition
Initiator	The character that started the SE
Receiver	The character that was the target the SE
ThirdAgent	An optional third participating character
Animation	Returns an Animation
Outcome	How the receiver will respond to the SE

Table 4.2: Abbreviated CiF Social Exchange

4.5 Social State

The **Social State** contains the information about the social environment between characters.

- **Status** represents how NPCs feel; Captures the character's mood. It represents by temporary values that are the result of a social exchange;

  There is a list of the Status, also known as Emotions, where it can influence the character's behavior, depending on their intensity. These statuses can be seen in Table 4.3, created based on Traits. For instance, if the character A feels more introverted or tired than the character B, so it has less motivation to start a Social Exchange than the character B.

Status Name	**Function**
ProSocial	The need that the character feels to have positive SEs. It will depend on character's Traits.
AntiSocial	The need that the character feels to have negative SEs. It will depend on character's Traits.
Anger	It will check how much Angry a character is. It will depend on rejected SEs and increases the volition to perform negative and hostile SEs.
Introvertedness	It will decrease the volition to perform any SE.
Tiredness	Increases periodically, and each time a character starts an SE as the initiator.

  Table 4.3: Short description of the Status.

  By default, Status will gradually decay over time. The *ProSocial* and *AntiSocial* status will increase with passing of time and can be decreased by successfully performing positive and negative SEs, for example the *Compliment* and *Bully* SE (tendency to have a talk with a friend or to bully an enemy).

  The Tiredness status will increase when the initiator starts a new Social Exchange to perform, and get close to the receiver to perform the interaction itself. If it is not doing any SE, or after finishing a SE, the character will be resting and the status will be decreasing. When the character is resting, it won't be able to start any Social Exchange.

- **Social Network Beliefs** are strong associations between the characters and the Player;

  Contains the non-reciprocal and private feelings that one character feels towards another character, being *Friends* or *Dating*. There is one Social Network Belief structure for each pair of characters who are in one social exchange, encapsulating the relation's name, intensity and NPCs involved.

If NPC A and NPC B don't have any relation, and they interact with each other, a new Social Network Belief entry is created. If already exists one, so it's updated with the new value.

Each NPC will have also their own SocialNetworkBeliefs, who will contain the beliefs of relations about others NPC with itself and between themselves. The belief is private, (for example, NPC A can think that the relation between NPC B and NPC C is 5 but if B and C performed any social exchange that NPC A didn't see, B and C will increase their relation to 6 and A will continue to think that their relation is 5).

Both memories and SocialNetworksBeliefs influence the final volition when the character it's calculating which social exchange performs next and with whom.

4.6 Microtheories

Microtheories (MT) are sets of Influences Rules that share a precondition, and are used to better organize sets of related Influence Rules. Some examples of Microtheories used are:

- *Dating IR*: if the characters are dating, it is available to perform SEs with *Romantic* intention and *Hostile* intention, depending on the traits and status of each one.
- *Friends IR*: if the characters are friends, it is available to perform SEs with *Positive* intention and *Negative* intention, depending on the traits and status of each one.

4.7 Trigger Rules

Trigger Rules are rules that provoke secondary effects, applied after a Social Exchange has ended, either by failing or succeeding.

For example: if NPC A hears NPC B insulting NPC C, it is expected that their relation (between NPC B and NPC C) will decrease, and, NPC A can decrease also the Friendship for NPC B.

On Social Exchanges with Romantic intention, for instance, if NPC A flirts with NPC B, who is dating with NPC C, one of the trigger rules will fire and then the NPC C will try to intimidate NPC A.

There is also an *Admiration* SE who only is performed when the Player's Clan level up. The characters around will notice it and one of them will start the interaction.

The Trigger Rules take advantage of the Status: instead of just checking if a character is *Angry*, we are able to check how *Angry*, with the status *Anger*, the character is and create rules that take this distinction into account. These values are changes at the end of SEs, depending on the Outcome.

4.8 Summary

The system architecture used as the basis for our model is the Comme il Faut architecture, already used before on some popular games such as Skyrim [16] and Conan Exiles [25]. It is a system designed to be able to generate social behavior for social agents.

On the original CiF, the Social Exchanges (being successful or not) have consequences once executed, and, the effects of each Social Move change the Social State, leading to a new cycle of the model. At any moment, unexpected events and consequences can occur, and these are handled by the Trigger Rules, which upon triggering may also affect and change the Social State. These components will help to make the NPCs more believable and to increase the user experience. The player must be able to "feel" the NPCs seeing what is happening in the scenario, and each one reacting in a different way, depending on their personalities and goals.

CIF-BL was based on CIF, but there are differences that were implemented. For instance:

1. a binary outcome, similar to CiF-CK instead of using it as implemented on CiF-EX, which will be discussed forward;

2. a centralized component, that will control how many Social Exchanges will be able to perform at the same time, and when they are initialized and finished.

   CiF-EX has a cooldown on CiF cycle to prevent the interactions happen all in a row, but on CiF-BL that cooldown was not implemented;

3. Meanwhile, both CiF and CiF-CK represented status as binary, a character was under a certain status or not. On CiF-EX, each status is represented by a value between 0 and 10, which allows for a greater detail when describing a character's disposition. Our model was inspired on CiF-EX.

The user is an entity that interacts with the CiF components. In both videogames that CiF-CK and CiF-Ex were implemented (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. But in CiF-BL, as in the CiF-Ex, the user only will control one character, which will be referred as the *Player*. This change was proposed in CiF-CK and was kept for our architecture. From the perspective of the CiF-BL, the Player is just like another character and as the characters will want to interact with each others, they will also want to interact with the Player. Social Exchanges can be initiated by the Player just like a NPC would, and pnce, the Social Exchange ends, the NPCs will create a new Memory summarizing the SE performed. The trigger rules are not calculated for the Player, since they have no internal attributes to be modified.

5
Implementation

Contents

In this chapter, we take a deeper look into how CiF-BL was implemented and integrated in Bannerlord. Features, challenges and difficulties will be discussed. Unfortunately, one of the initial problems was that there was no documentation about the source code of the game, and furthermore although Taleworlds does state that they support modding, the code is not directly accessible. Therefore, in order to understand how to implement and integrate our mod, a reverse engineering process was used to extract the code from the binary files. By doing this decoding process, we were finally able to access the game's classes and methods, and to also understand the game's architecture. Our developed mod was fully implemented in C# using *Visual Studio* to create, debug and package the new developed code.

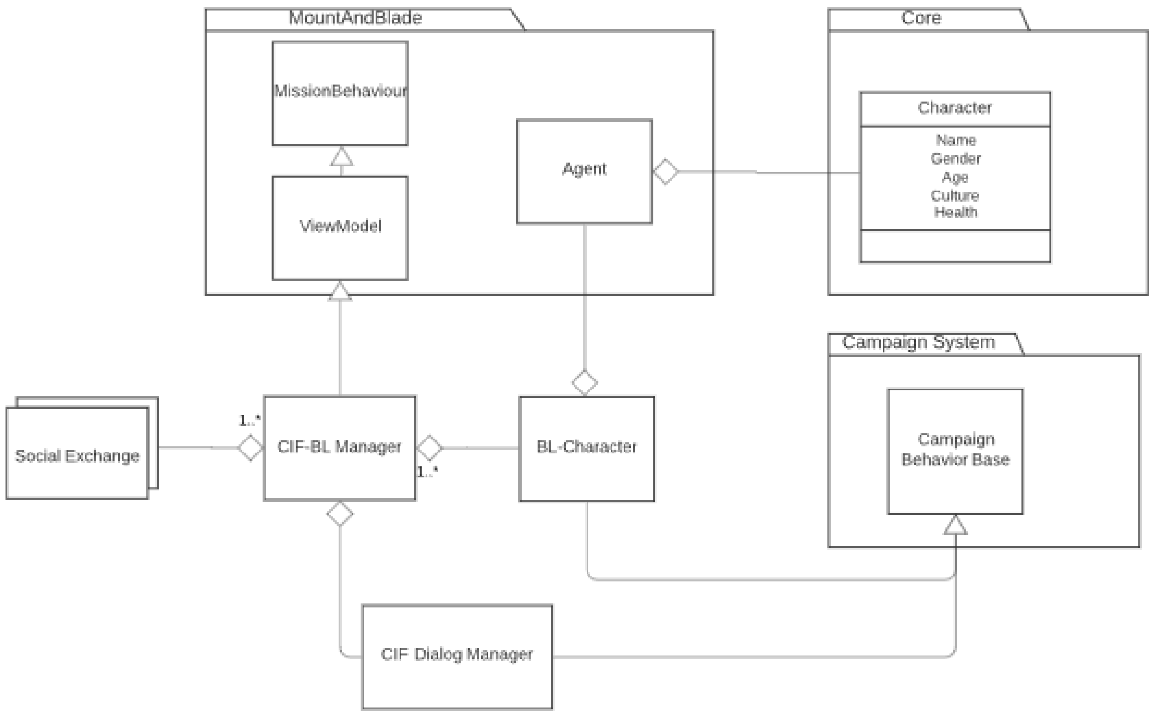


Figure 5.1: Connections between CIF-BL and Bannerlord

The connection between the CIF-BL model and the Bannerlord's game, as shown in the Figure 5.1, is possible by creating a new mission behavior and a new Campaign Behavior, respectively CIF-BL Manager, which is responsible to manage the model, and CIF Dialog Manager, which is responsible for the CiF-BL Dialogs. Both were added to the game's collections at startup, MissionBehavior's and CampaignBehavior's collection. That leads for when the game is running its own system, it runs also the implemented CIF-BL model.

Most of the logic on the CIF-BL model is concentrated on some classes: *Social Exchange*, *BL-Character* and *CiF-BL Manager*. The *CiF-BL Manager* manages and functions as the centralized controller, it runs the CiF-BL cycle who contains all the *BL-Character* references and events about which Social Exchange is initialized. The *BL-Character* stores most of the character's social attributes and the management to the NPCs, for example the movement. The *Social Exchange* handle the computational part of the architecture.

During the current map's initialization, the *CiF-BL Manager* creates the *BL-Character* instances. After, *CiF-BL Manager* handles and controls the BL-Characters, their desires and, finally, their *Social Exchanges*.

5.1 Social Exchange

As discussed before on the section 4.4, a Social Exchange has some properties that are discussed with more technical detail on this section.

Term	**Function**
Intention	Intention_Enum
SocialExchange	SocialExchange_Enum
InfluenceRules	Float
Initiator	Agent Reference
Receiver	Agent Reference
Animation	Returns an Animation
Outcome	How the receiver will respond to the SE
Volition Function	Float

Table 5.1: Brief Social Exchange Description

As it shown on the Figure 5.1:

- **Intention** is a custom Enumerator. The values can be Positive, Negative, Romantic, Hostile and Special (AskOut and BreakUp SEs).

  These Intentions were created with the purpose to have one specific to increase and decrease the relation's intensity when the characters are *Friends* or *Dating*. The Special Intentions will be allowed to change the relation's name from *Friends* to *Dating* and vice-versa.

- **Influence Rules** are defined on code and generated when the SE is initialized.

  They will check each character's *Trait*, *Status*, the culture's relation that the participants' culture has between both, and an additional "Initial Value" parameter who is the current relation that the initiator has with the receiver.

- **Initiator** is the NPC who will start the interaction with the receiver. Initiator Volition is used to prioritize what SE to execute.

- **Receiver** is the NPC who will be interacted by the initiator. Receiver Volition is used to assert how the receiver will react to the SE.

- **Animation** will return a Bannerlord's animation, to both initiator and receiver, that matches the current Social Exchange. The animations are movements and actions that the character's model performs when they are returned. They are stored in one XML file inside the game folders, and are re-used by the CiF-BL model.

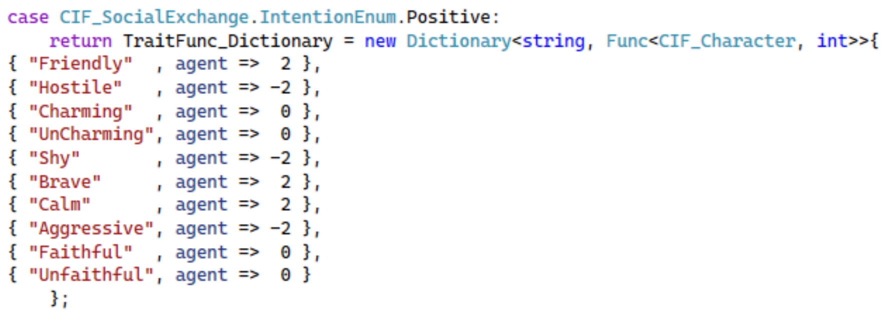
- **Outcome** is an integer that represents how good or bad is a social exchange. A *Positive Outcome* corresponds to any number bigger than 0, and, a *Negative Outcome* when it's equal or lower than 0.

- **Volition Function** has the functionality to get a character's volition, to start or react, to one Social Exchange.

  This Function has 5 attributes that influence on the final volition:

  1. **Traits**: Each trait has different weights when computing the volition, for each intention, as shown in the Figure 5.2 with the *Positive* Intention's example. When an Intention is Positive, *Friendly* trait will have more weight, unlike to *Hostile*.

     For each *Intention*, it is returned a dictionary with the values for each trait. Depending on whether or not the character has the trait, the respective value will be added to volition or not.



```
case CIF_SocialExchange.IntentionEnum.Positive:
    return TraitFunc_Dictionary = new Dictionary<string, Func<CIF_Character, int>>{
{ "Friendly"   , agent =>  2 },
{ "Hostile"    , agent => -2 },
{ "Charming"   , agent =>  0 },
{ "UnCharming", agent =>  0 },
{ "Shy"        , agent => -2 },
{ "Brave"      , agent =>  2 },
{ "Calm"       , agent =>  2 },
{ "Aggressive", agent => -2 },
{ "Faithful"   , agent =>  0 },
{ "Unfaithful", agent =>  0 }
    };
```

Figure 5.2: Example about the Trait's value with Positive Intention.

  2. **Status**: depending on the status intensity, it has different weights to be added to the volition.

For instance, if *ProSocial* Status has an intensity between 0.5 and 1.5, it increments 2 for Social Exchanges with *Positive* Intention. But if it has an intensity bigger than 1.5, it increases 4 to the volition.

Working similarly for the other Status, like the *Anger* Status that increases for *Negative* and *Hostile* Intentions, decreasing by other side, for *Positive* and *Romantic* Intentions;

3. **Culture**: may affect also the character's volition. Depending on if a participant's culture is friendly, unfriendly or neutral with the other participant's culture, it increases 2, -2 or 0, respectively, to the character's volition;
4. **Memory**: for each time that a SE was performed with a specific receiver, it decreases the volition.

   It also decreases if a NPC remembers the SEs performed previously, for instance, if it stops dating with a character, until it remembers, it doesn't start a SE to restart dating again with that character;
5. **Conditions**: Some logical rules that are verified and apply. Each Social Exchange has specific rules to validate. For instance, if a character is dating another character, it doesn't ask to start dating again;

The Social Exchanges that were implemented, with the respective Intentions, can be seen on the Table 5.2:

Intention	**Social Exchange**
Positive	Compliment, GiveGift, Admiration
Negative	Jealous, FriendSabotage
Romantic	Flirt
Hostile	Bully, RomanticSabotage
Special	AskOut, Break, HaveAChild

Table 5.2: Short description of the Intentions & Social Exchanges

5.2 BL-Characters

A BL-Character is a component of the CiF-BL architecture. It is a structure that is associated with the NPC on game, that took advantage of the already existing character's class, named *Agent*.

The **Agent** class, defined by the game, contains all the information about the character (Name, Gender, Age, Culture, etc.). The majority of the content that it contains in game it is stored on this object.

It was created a new class to extend the original Agent's class, which serves as base, with the CiF-BL properties, named *BL-Character* and added the specific attributes for each one.

The connection between both classes was possible by using a BL-Character's variable, named *AgentReference*, that contains all the specific information about each one, referencing to a concrete character on the game, as shown on the Figure 5.1.

In short, a *BL-Character* has defined a variable named *AgentReference* who makes the connection between the *BL-Character* and the *Character* defined on the game.

There are a few essential attributes on the BL-Characters and displayed in the Table 5.3 that has an overview of this structure, for a detailed view see Table A.3.

Element Name	Implementation
CiFReady?	Boolean
Busy	Boolean
IsInitiator?	Boolean
Target	Agent
Countdown	Float
TopSE	List of Social Exchanges
Social Network Beliefs	List of SNBelief_Object
Social Exchange Memory	List of MemorySE_Object
ItemList	List of Strings
StatusList	List of Strings
NextSE	CiF_SocialExchange_Enum

Table 5.3: Abbreviated CiF-BL Character

- **CiFReady?** is *True* if the character is ready to start a SE, corresponds to *Busy* variable and *Tiredness* status. It is False for a period of time after the character has a SE, it functions as the On Countdown variable.
- **Busy** is *True* if the character is busy interacting with other character or Player;
- **IsInitiator?** set the character as initiator or receiver.
- **Target** will contain the receiver's reference to interact.
- **Countdown** is how much time the character will be not allowed to start a new Social Exchange.

This management is made by the centralized controller *CIF-BL Manager*, as discussed in Chapter 4, avoiding the same NPCs always keep interacting with each other and interrupting the Player constantly.

- **TopSE** it will store the Social Exchanges with higher volition when generating a new SE to perform.
- **NextSE** will contain the reference for the character initialize next Social Exchange as initiator, picked randomly from *TopSE*.
- **Social Exchange** is the variable that represents the Social Exchange. For more information, see 5.1. Each SE's participant will have defined the current Social Exchange on going to determine how to act/talk.
- **Social Exchange Memory** stores memories of SEs if the NPC is less than a defined distance to the SE initiator. One memory contains strings to store the names of the Social Exchange, Initiator and Receiver;
- **ItemList** it will hold the information about the items who are in the character's inventory. Having at least one item, allows the character to perform the *GiveGift* SE.
- **StatusList** will contain all the information about the character's status.
- **Social Network Beliefs** are matrices that represent the beliefs of Current Social Network values for different characters.

  Each matrix will contain strings for the characters' names, Integers to store their ID's, String for relationship type (*Friends* or *Dating*) and Integer for relationship's intensity ranging from 0 to 10.

5.2.1 Persistence

In Bannerlord, characters will only spawn on the world when the Player enters on one place (e.g.: if the Player enters in Town A, the NPCs on that town will be created and added to their defined locations). When the Player leaves the place, the characters in that place will be destroyed and the information will be saved on the game's save file for persistence (e.g.: if the Player leaves Town A and enters Town B, the NPCs' characters on Town A will be destroyed and the new NPCs on Town B will be created).

For character's data persistence, the CIF-BL Manager checks when the Player enters in one location if it's the first time or not. It will check also how many characters will be in there, and it will extend them by creating and attaching to each one their respective generated personalities. All the CiF-BL information about their personalities, items and memories, will be stored on one external file *data.json*, created also by the script, using JSON language. With this way, our model will load and save data as the game does without conflicts.

There are a few points who deserve more attention on this save/load system. The process of loading the characters with the correct information, made by the game, it's done by using a list. That means the character's order it will be the same, starting always by the Player's character, companions, lords, and, common citizens (the characters who doesn't have any role on game).

This is important to know for two main reasons:

1. The Social Exchange's collection and Character's collection are both static. The game creates the characters always in the same order, then the CiF-BL Manager extend them and create the BL-Characters also in the same order.

   This leads to have the same initial characters having high probability to initialize the initial Social Exchanges, reaching the maximum ongoing SEs quickly and don't give opportunity to the last characters on the collection.

   This is the reason that the characters store the Social Exchanges with higher volition and pick one to perform, instead of perform the first SE with more volition, as discussed in Chapter 4.

2. When spawning the characters on one location, there will be more agents created on the day than on the night.

   That means, when entering on one location for the first time during nighttime, traits are only generated for the available NPCs. If later the player comes back to the same location during the day, the file with the saved information for each character will be incomplete, because there are NPCs who were not initialized with CiF-BL attributes.

   To avoid having characters without personalities, the CiF-BL Manager will always check if there are any new characters on a particular location, and if there are, it will generate new traits for them and save the new information.

5.3 Dialog System

The current Dialog System implemented by the game can only be used for interaction between the player and NPCs, and has mainly an instrumental functionality, for example to accept and complete quests from quest givers, to start trading with the merchants, and, to start battles with other armies.

We reused the current system to add additional phrases and options when the interaction occurs between Player and NPC, as usual. But to develop a system for social exchanges between NPCs, a new dialogue system was adapted and implemented, reused from another system in the game (named *MissionNameMarker*, it will be discussed later).

There are 3 different types of interactions that can occur in CiF-BL model:

1. Player as initiator and NPC as receiver;
2. NPC as initiator and Player as receiver;
3. NPC as initiator and NPC as receiver;

5.3.1 Player and NPC Dialog

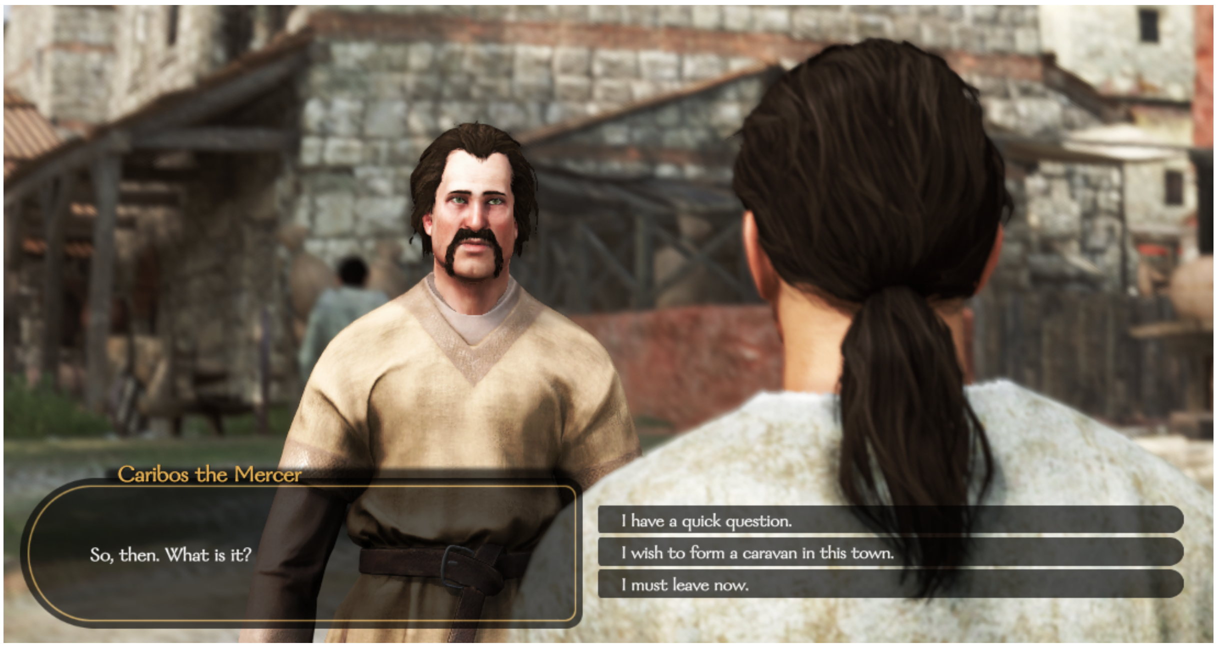
This is the conventional form of interaction already supported by the game. The player takes the initiative to go and talk to a particular NPC. The game's original dialog system was reused to implement the interactions between the Player and the NPCs controlled by the CiF-BL architecture.

The game adds dialogs and Player's choices by using the *ConversationSentence* object's class from *CampaignGameStarter* class. The *ConversationSentence* will contain both methods: *AddDialogLine*, to add sentences to character, and *AddPlayerLine*, to add options to Player.

In the game, User Interface (UI) containing sentences and choices only appear when the Player interacts with a NPC, as shown in the Figure 5.3. This is the first interaction that was already implemented by the game.

To allow the player to perform social exchanges, which can be perceived as such by other NPCs, the dialog system was extended with CiF-BL Dialogues - dialogues with their own conditions and consequences. The CiF-BL sentences are defined in one external json file. They are only loaded and attached to the correspondent NPCs when the characters are spawned, as the games does by default. The CiF-BL dialogs will be the available Social Exchanges, and they will appear along with the game options, as shown in the Figure 5.4.

When it is a NPC starting the conversation, the NPC (initiator) approaches the user's character and the User Interface opens without the Player to start the interaction. This interaction increases the player's feeling that the NPC is taking the initiative of interaction. When a character initializes a conversation with the Player, depending on the social exchange, it says something and different options to react are displayed, as shown in the Figure 5.5.


Caribos the Mercer

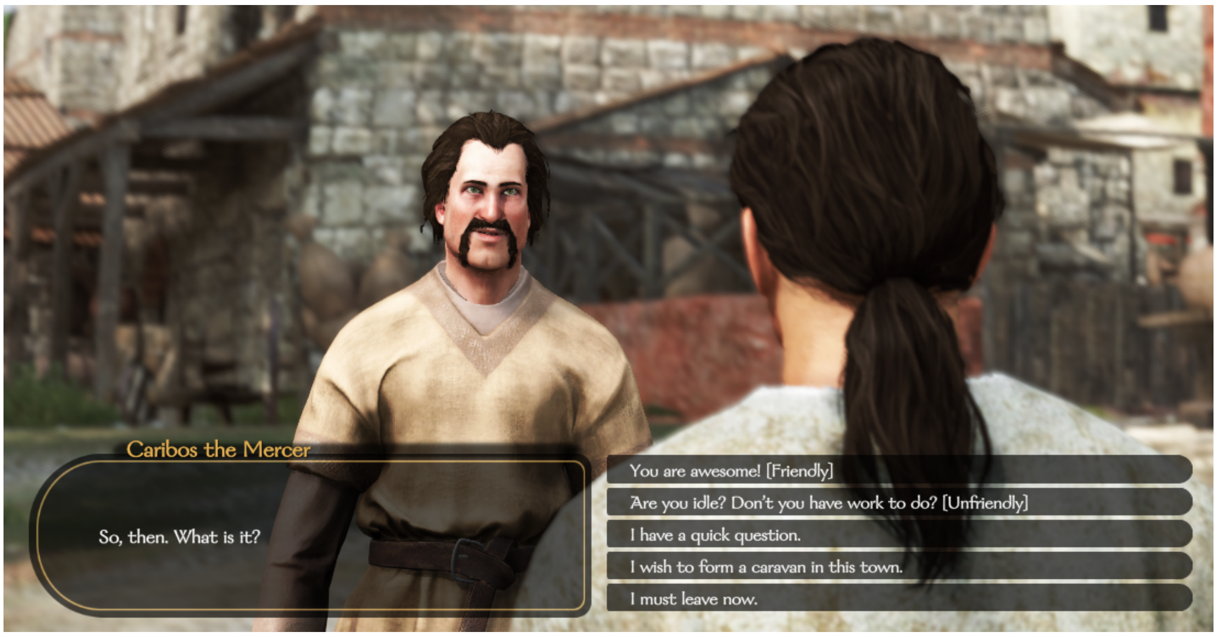
So, then. What is it?

I have a quick question.

I wish to form a caravan in this town.

I must leave now.

Figure 5.3: Example of Player interacting with NPC by default.


Caribos the Mercer

So, then. What is it?

You are awesome! [Friendly]

Are you idle? Don't you have work to do? [Unfriendly]

I have a quick question.

I wish to form a caravan in this town.

I must leave now.

Figure 5.4: Example of Player interacting with NPC using CiF-BL.

Each dialog line has a few parameters that are essential when adding to the game:

- **ID**: Identifier.
- **Input Token**: It will be used to identify the dialog to display next. Normally, on the beginning of one conversation with a character, the token will be defined as *start*.

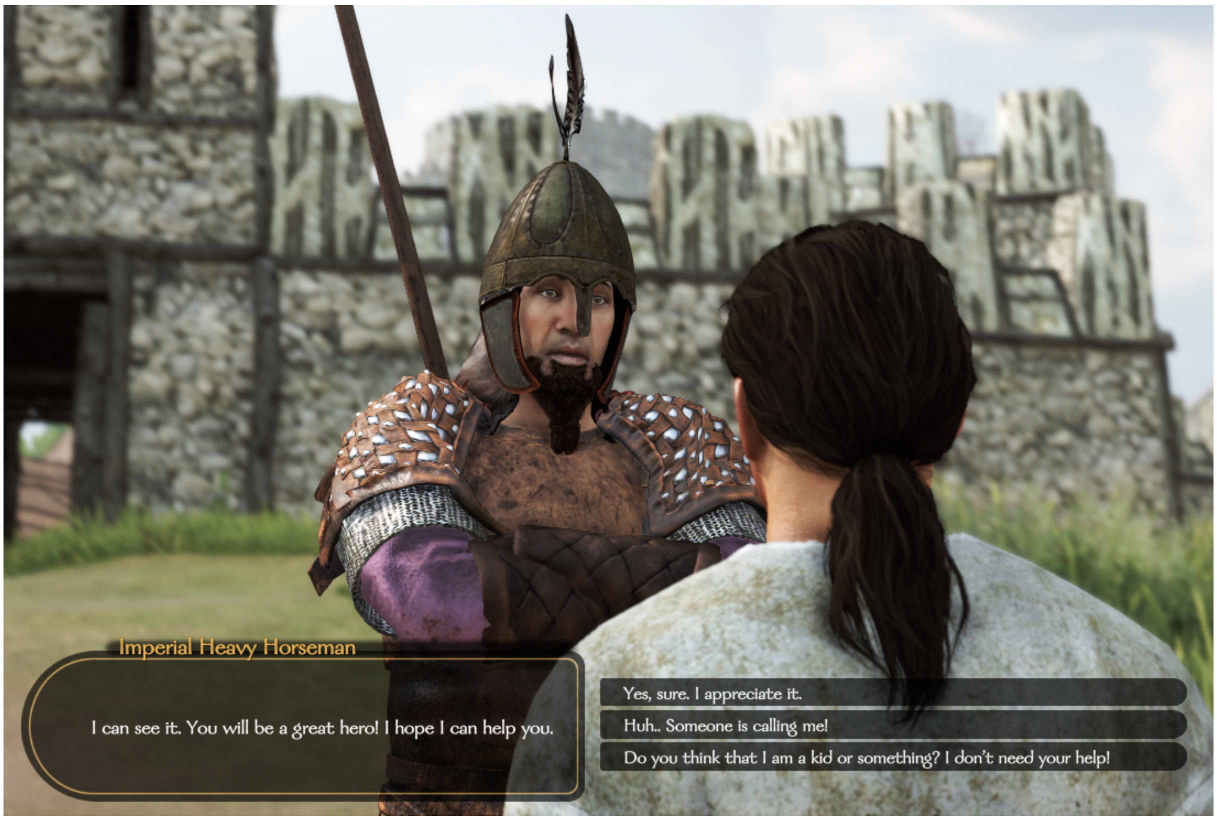

Imperial Heavy Horseman
I can see it. You will be a great hero! I hope I can help you.
Yes, sure. I appreciate it.
Huh.. Someone is calling me!
Do you think that I am a kid or something? I don't need your help!

Figure 5.5: Example of and NPC interacting with Player using CiF-BL.

ID	string
Input Token	string
Output Token	string
Text	string
Priority	int
Condition	OnConditionDelegate
Consequence	OnConsequenceDelegate

Table 5.4: Parameters for dialogues' creation.

The Player's options, PlayerLines to keep going with the interaction, will need to have the same *inputToken* as the DialogLine displayed.

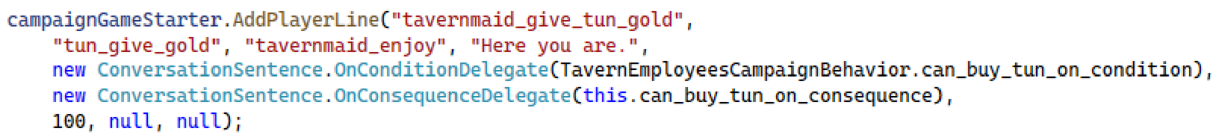
To be added a dialog as an option to the Player, it will depend also on the character (for instance, if we want to add a specific dialog to Hero Characters, the input token must be *hero_main_options*, and the PlayerLine will appear available to choose when starting a conversation with a Hero).

- **Output Token**: It will redirect the current sentence to the next one. If the conversation it’s to end, the Output Token is *close_window*.

Normally, on the dialogues flow, it is used an input token to start the conversation, who will occur depending on the Player’s choices. Each Player’s option will have their own output tokens, and depending on what the player chooses, it will redirect to the next phase of the conversation. The Output Token from one option it will be the Input Token from the next dialog and vice-versa (except when they are *start* or *close_window*).

- **Text**: It will be the message spoken by the NPC and the message of the Player’s option.
- **Priority**: The NPC will display the dialog with the higher priority, and the Player will have their options ordered. It was used in rare exceptions on the game, without the need to change the default value.
- **Condition**: It is a delegate that returns a boolean to check when the dialog must or not appear on the interaction (e.g. if the Player is doing a quest about to get some items, the sentence of deliver the items only appears once the Player has them). Default value is null, which is no conditions applied to phrases. IN CiF-BL, these type of conditions are used to condition existing social exchange options it is used, for example, to display or not the options to interact friendly or unfriendly with the NPCs, depending on their relation.
- **Consequence**: It is a delegate that doesn’t return any value. Used to trigger a specific event, when the Player chooses an option on the interaction (e.g. when buying items through dialog, recruiting a caravan, etc). Default value is null. On CiF-BL, it is used to Increase and Decrease the relations with the NPCs depending on the Player’s chooses.

On the figure 5.6 its displayed a dialog line with a condition and a consequence.



```
campaignGameStarter.AddPlayerLine("tavernmaid_give_tun_gold",
    "tun_give_gold", "tavernmaid_enjoy", "Here you are.",
    new ConversationSentence.OnConditionDelegate(TavernEmployeesCampaignBehavior.can_buy_tun_on_condition),
    new ConversationSentence.OnConsequenceDelegate(this.can_buy_tun_on_consequence),
    100, null, null);
```

Figure 5.6: Dialog Line with Condition and Consequence Example.

Following the example on Figure 5.7, when the Player goes to interact with a character (e.g.: *Townsman* or *Townswoman*), the character asks if there is something that the Player needs. This first Dialog Line contains as *inputToken start* to define the sentence, the first phase of the conversation. The *outputToken* defined as *town_or_village_player*, are the options available to the user to select from, so that the player has the opportunity to guide the conversation as desired.

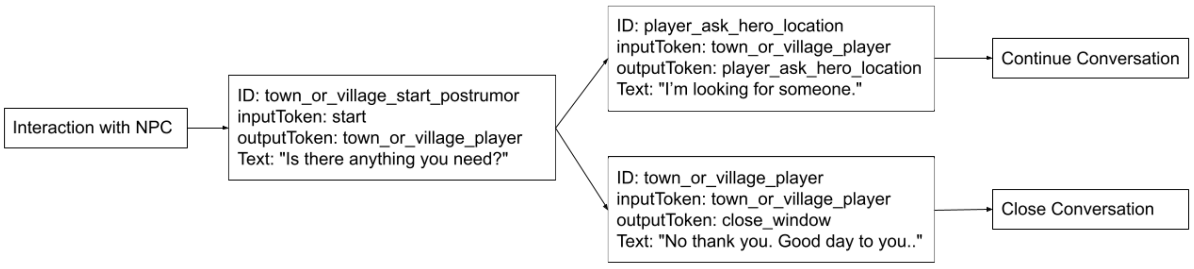

Interaction with NPC
ID: town_or_village_start_postrumor
inputToken: start
outputToken: town_or_village_player
Text: "Is there anything you need?"
ID: player_ask_hero_location
inputToken: town_or_village_player
outputToken: player_ask_hero_location
Text: "I'm looking for someone."
Continue Conversation
ID: town_or_village_player
inputToken: town_or_village_player
outputToken: close_window
Text: "No thank you. Good day to you.."
Close Conversation

Figure 5.7: Simplified Dialog's Logic View on Game.

Each dialog line, with its own purpose, conditions and consequences, have as inputToken the previous dialog's outputToken.

If the Player was never introduced to a NPC (as initiator or receiver), and it's the first time who they are talking, so the introduction game's dialog is displayed. It is one of the exceptions made by the own game that the sentences have higher priority to appear.

The Player is only able to perform a social exchange with one intention to a NPC once a day, a rule implemented to not be able to increase quickly the relations with the other characters once that higher the relation, better are the rewards from some characters.

Meanwhile, the Player's dialogs to perform interactions with other characters, it is chosen randomly from a defined group of alternatives, created on the external file.

This will avoid the Player to be constantly interacting with the Lords and increasing, easily, their relation, and the feeling of repetition.

5.3.2 NPC and NPC Dialog

For the interaction between NPCs, the current dialog system is not efficient.The idea to make the Player capable to listen what characters are talking, was possible by adding the feature to be possible to see their conversation, and respectively, interacting with each other.

To implement it, and reach the previous idea, it was necessary to use a game feature that is used to show the relevant locations and targets on the current map. For instance, in Figure 5.8 we can see 3 makers - two above the NPCs head, and one in the entrance to the tavern. This functionality is implemented in the base version of the game by the *MissionGauntletNameMarker* class, who will create and manage a *MissionNameMarker* object's class.

For the dialogs between characters, using CiF-BL, it was developed the classes *NPCDialog-*

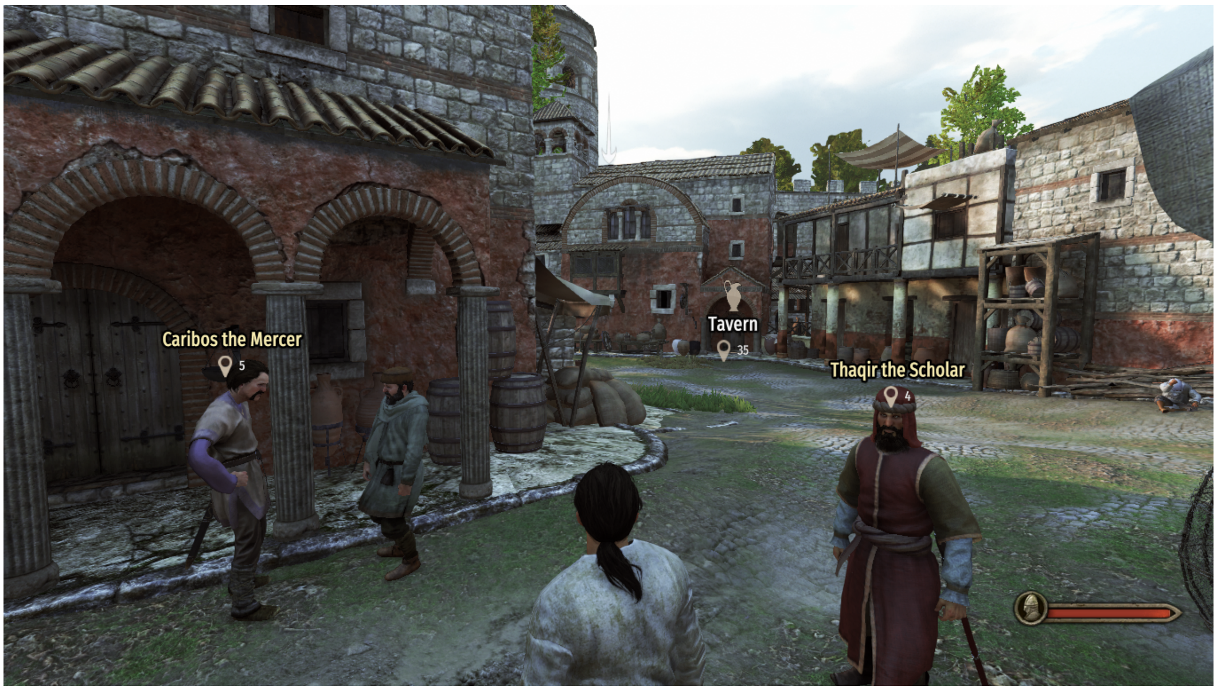


Figure 5.8: Example MissionNameMarker system.

Marker and *CIF-BL Manager*, a modification respectively of *MissionGauntletNameMarker* and *MissionNameMarker* classes.

The *NPCDialogMarker* is responsible to initialize and finalize the dialog user's interface, when the Player loads a new/saved game. The *CIF-BL Manager* manages and handles the CiF-BL Cycle, as discussed before in Chapter 4.

Instead of the character's name, it displays what a NPC is saying when interacting with another character. As shown in the Figure 5.9, there is possible to see that are ongoing two different conversations between different characters.

For the positive and negative social exchanges, the color of the sentences are, respectively, green and red, as shown previously on the figure 5.9. The time that the NPC's message is displayed on the screen, it is possible to be defined by the user itself, on the configuration file, or it keeps the value by default (3 seconds).

Each culture has their own specific phrases for each social exchange (when starting, accepting or rejecting a SE), as displayed on the figure 5.10. The Player when interacting with specific NPCs has also different sentences depending on the character itself (Townsman, Lord, Tavernkeeper, etc).

The dialogues used by the CiF-BL system, are created and stored on external Json files, where they are imported to the game, as explained before.

On the sentences, some keywords are dynamic, which means, their value can be changed on

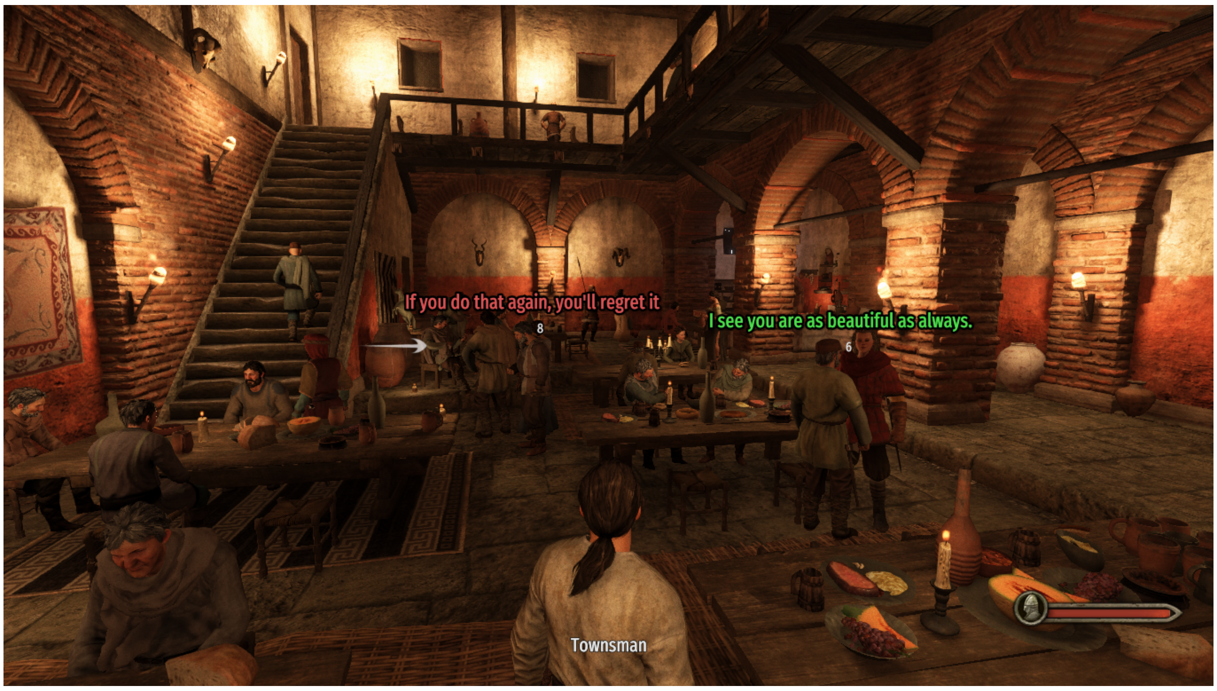

If you do that again, you'll regret it
8
I see you are as beautiful as always.
6
Townsman

Figure 5.9: Example of NPCs interacting with each other using CiF-BL.

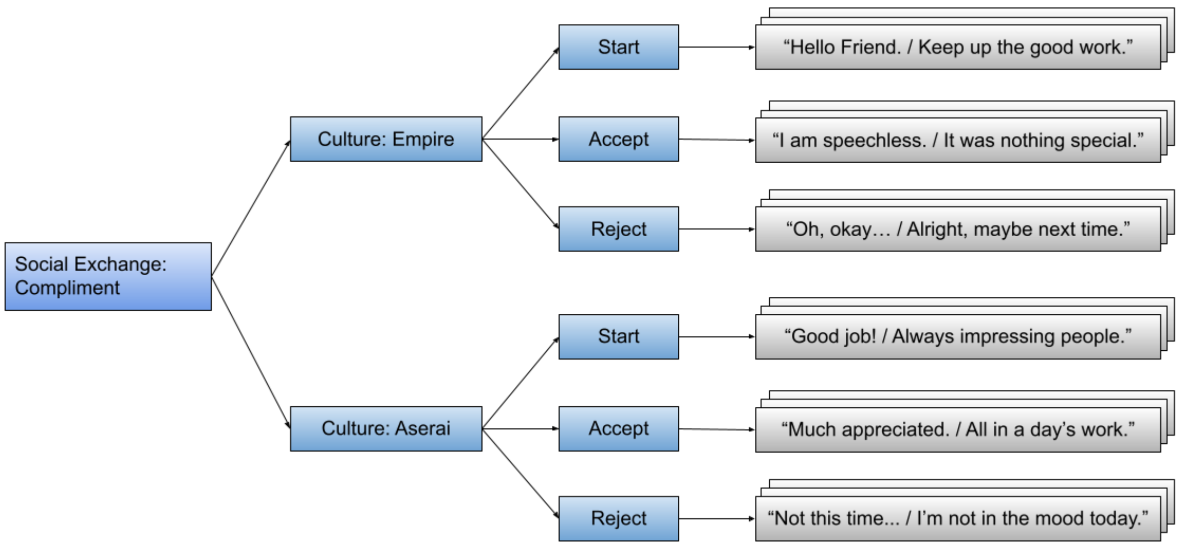

Social Exchange: Compliment
Culture: Empire
Start
"Hello Friend. / Keep up the good work."
Accept
"I am speechless. / It was nothing special."
Reject
"Oh, okay… / Alright, maybe next time."
Culture: Aserai
Start
"Good job! / Always impressing people."
Accept
"Much appreciated. / All in a day's work."
Reject
"Not this time... / I'm not in the mood today."

Figure 5.10: Example about how SE and character's culture influence the dialogs.

runtime. For example, if it's happening the GiveGift SE, the word defined on file as "{ITEM}" is replaced by the item's name that the initiator is offering to the receiver. Also, the color of the sentences can change to, respectively, to green or red, depending on the social exchanges being more positive or negative.

The conditions and consequences are converted from the file (string type) to the game

(OnCondition and OnConsequence delegate types) by using dictionaries. Each specific string corresponds to a specific condition and consequence. On Tables A.5 and A.6, there are an overview about, respectively, the conditions and consequences implemented.

The dialogues were also added to NPCs when the Player is on the battlefield fighting against other army, as shown in the Figure 5.11. The characters on the Player's team have their phrases with the yellow color and the enemy team has their sentences with red color, to know from which team is who is speaking.

These sentences are not depending from Social Exchanges, but if the army is winning or losing the battle. Each character has a countdown to display a message, and it is defined also a limit of *speakers* (as the maximum of SES).

These definitions can be changed by the user, and if it's preferable to disable or not the battle's dialogs, on the configuration file that it's available along with the mod.

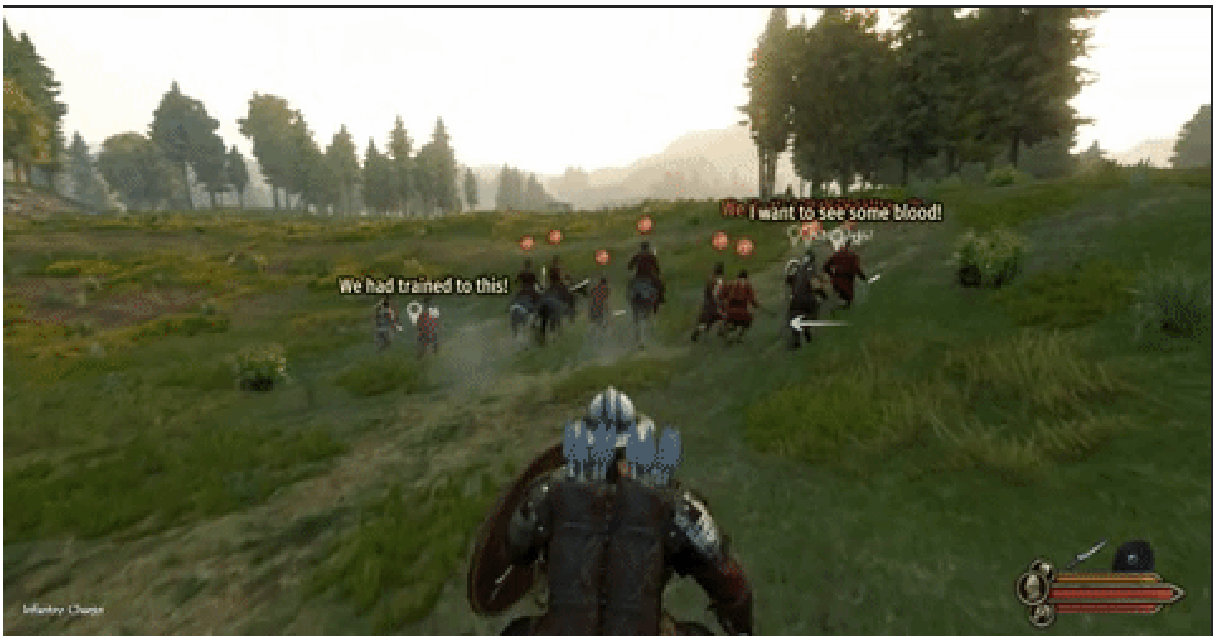


Figure 5.11: Example of NPCs during battle using CiF-BL.

5.4 CiF-BL Cycle

On this section, we describe how the above-mentioned components interact with each other when executing a CiF-BL cycle.

5.4.1 Decision Making

The **CIF-BL Manager** does the actual decision-making for the BL-Characters. It makes possible the connection between the CiF-BL Cycle and the dialogs, when the Player is involved in the interaction.

- In our mod, each social exchange has 3 scenarios available:
  - Successful Social Move
  - Failed Social Move
  - Social Move with Player
- All scenes have at least 3 phases:
  - **Desire Formation** phase, where the initiator calculates and chooses the next social exchange to perform, with a respective receiver.
  - **"Go-to"** phase, where the initiator moves towards the receiver (the initiator has the desire to interact with a target, but the conversation between both didn't start yet).
  - **Perform the Interaction** phase, when the initiator is close enough to the receiver, it performs the interaction.

When the Player is not involved, NPCs expect an immediate response as a consequence. There is an additional phase: the "Response" phase, when the receiver responds to the initiator declaring if it accepts or rejects the social move.

After calculating the volition to start performing a Social Exchange, the initiator begins going to the receiver's position. When they are near each other, the conversation starts between both and the social exchange itself.

When the Player is involved, it is displayed the dialog's UI, containing the choices to accept or reject the social exchange initiated by the character, or, to start one with it.

For each Social Exchange, there is one animation associated and different dialogs lines, depending on their role (initiator or receiver) and on their culture (for dialogs). Also, some may also require additional steps, for example if GiveGift Social Exchange is happening, it is required to transfer an item from the initiator's inventory to the receiver's inventory. There are also Social Exchanges that only are performed when happening in specific events on the game, as discussed previously.

On the conversation, they will talk by turns (starting by the initiator) and each one has displayed the sentence during a few seconds, and they can have longer or shorter conversations depending on the number of sentences available.

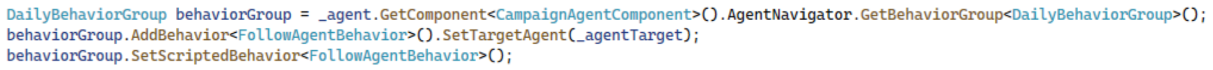
5.4.2 Desire Formation

Before this phase starts, the algorithm checks if the number of ongoing SEs is less than the defined maximum of simultaneous SEs. If the maximum has not been reach, there's no cooldown on, so the *Desire Formation* process is started if there is available, at least, one NPC.

CiF-EX has a cooldown on CiF cycle to prevent the interactions happen all in a row, but on CiF-BL that cooldown was not implemented. The only one who exists is the NPCs cooldown (depending on the traits) and when they are recovering their energy (depending on the status: Tiredness).

5.4.3 "Go-to" phase

After the initiator generates a desire, having a social move to perform with another character, it moves closer to the receiver. To give this behavior to move a character from one position until the receiver's position, was used an already implemented method who adds a specific behavior: *Follow*. This behavior initializes *FollowAgentBehavior* to initiator setting as target, the receiver. It is simple to make a NPC walk to the target location, by using the method shown in the Figure 5.12.



```
DailyBehaviorGroup behaviorGroup = _agent.GetComponent<CampaignAgentComponent>().AgentNavigator.GetBehaviorGroup<DailyBehaviorGroup>();
behaviorGroup.AddBehavior<FollowAgentBehavior>().SetTargetAgent(_agentTarget);
behaviorGroup.SetScriptedBehavior<FollowAgentBehavior>();
```

Figure 5.12: Follow Agent Behavior.

The initiator soon is close enough to the receiver, and it starts performing the interaction. This distance, between both characters, is a defined value of 3 meters is the same distance to make the Player available to interact with characters (defined by the game itself).

5.4.4 Perform the Interaction

Initiator has a social exchange containing the respective intention to perform with receiver, generated during the *Desire Formation* process.

When initializing the interaction, the initiator imports the sentences available to say by using a dictionary, with all the dialogs preloaded by an external file on the initialization. Using the initiator's culture and the social exchange that it will perform, the NPC gets the sentences available to *start*. These sentences are used to say to the receiver, and, to display them on the User Interface to be visible to the Player during some time. From the receiver's side, when it calculates the volition to react, it also imports the sentences to speak and respond to the initiator.

These sentences depend on from the social exchange and from the character's culture, having the difference of *accept* or *reject* the SE, as receiver.

The characters will be speaking by turns during a specific time. The conversation happens, until both don't have more phrases to say, being updated the participants' Status, Memories, and Beliefs according to how it ended.

6

Evaluation

Contents

This thesis' goal was to create more socially engaging and believable NPCs, hoping it would lead to an improved user experience and enjoyment. In this chapter, we evaluate if the CIF-BL model implemented was indeed able to achieve the aforementioned objectives.

6.1 Procedure

Unfortunately, due to COVID-19 restrictions that occurred during the execution of this thesis, it was not possible to conduct a controlled in-person evaluation (which would be the ideal way to evaluate our work). Instead, a mod was produced and released online to the Bannerlords player and modding community. The only difference between the mod and the base version of the game was the CIF-BL model used to control the NPCs. The mod was called "Friendly Lords" and released in 18th of August 2021, both in NexusMods and in ModDB. The mod was advertised on social media, such as Reddit, on groups that were related to the game.

It is important to point out that all participants already had the Mount & Blade II: Bannerlords game installed, and had plenty of experience playing the original version of the game. This also means that there is very likely a bias of participants towards liking the Bannerlords game. Unfortunately, it was not possible to recruit participants outside of this community, since it would require us to offer a license to install the game.

Members of the community were asked to download and install the mod on their own computer and experiment the mod. A public discord server was created to provide additional information about the mod - and allow participants to talk with the developer. Participants in the discord server were asked to fill out a questionnaire after experimenting and playing with the mod.

When we deployed the mod, the game had the stable version 1.6.0 and the beta version 1.6.1, and it was released officially having compatibility to version 1.6.0. During the first week, the game released a new stable and a new beta version. This means that the version of the mod was outdated for players who were updating the game, and it was needed to update it to the newest version, to keep the current users playing with the mod, and attract new players. After a few days we deployed a new mod's patch supporting the game versions 1.6.0, 1.6.1, and, 1.6.2-beta.

It is possible to find the *Friendly Lords* mod on NexusMods the Mod's page[1] and the developed code on GitHub[2].

[1]https://www.nexusmods.com/mountandblade2bannerlord/mods/3230

[2]https://github.com/DavidCRicardo/Bannerlord_Mod_Thesis

6.1.1 Metrics and instruments

To evaluate the model, we focus in three main metrics: believability, quality of social interaction, and enjoyment. Believability can be viewed as a part of the player experience, who in general can be measured via reporting. The most direct method to evaluate believability is to ask to the players about their experiences [19]. To measure believability we used questions from the questionnaire for Believability proposed by Gomes [42].

Evaluating the engagement of social interactions (or quality of social interaction) is not as clear, as we could not find any questionnaire in the literature that would evaluate this directly. The closest questionnaire we could find is the *Social Presence*'s Scale first developed by Gunawardena and Zittle [43]. This scale was designed to evaluate Social Presence Theory [44], which according to Gunawardena represents "the degree to which a person is perceived as a 'real person' in mediated communication."

There was a strong requirement for making the questionnaire as short as possible, which made us only select two questions from each subscale of the social presence questionnaire proposed by Harms [45] (Co-Presence, Perceived Message Understanding, Perceived Affective Understanding, Perceived Emotional Interdependence, and, Perceived Behavioral Interdependence). The attentional allocation subscale was not used, since this subscale is more appropriate to evaluate scenarios with interactions only between two persons. We adapted the questions in order to make them more appropriate for our scenario, for instance: "I could tell how my partner felt" was changed to "I could tell how NPCs felt".

Finally, for the enjoyment metric, we used one question from the enjoyment subscale from the Intrinsic Motivation Inventory [46]. Questions related to these instruments were measured using a Likert Scale from 1 to 7. We asked the users to point how much they agreed with each sentence. The questionnaire also incorporates demographic questions.

Given that participants in the study already had experience with playing the game without the mod, we asked all questions (regarding believability, social presence, and enjoyment) for the two versions of the game: bannerlord without mod, and bannerlord with Friendly Lords mod[3]

In addition to the questionnaire (see Table B.1), we also collected data about the downloads and usage of the mod, and collected in-game data about the type of social exchanges performed by players. We also collected feedback from participants that were active in the Discord server.

Given that questionnaire was given only at the end of the players' experimentation with the mod, and we wanted to ask.

[3]Ideally, in a controlled setting, players would play version A of the game, then fill out a questionnaire about that experience, then play version B and then fill out a second questionnaire. Unfortunately, it was not possible to do so in our experiment.

6.2 Results

6.2.1 Impact and community feedback

On the first day, the mod gained a lot of attention after release. It immediately received many comments, however since members of the community hadn't played the mod yet, the comments were mostly remarks about how it was a good idea to improve the NPCs, and questions about how exactly the mod worked. Only a brief description of the mod was given, since we did not want to bias participants.

Two weeks after the mod's release, the Friendly Lords mod was displayed on the "Trending", "Most Downloaded (30 days)" and "Hot Mods" categories, as shown in the Figure 6.1.

At the writing of this thesis, players are still downloading, playing and giving positive comments about the mod, suggestions and ideas to improve it.

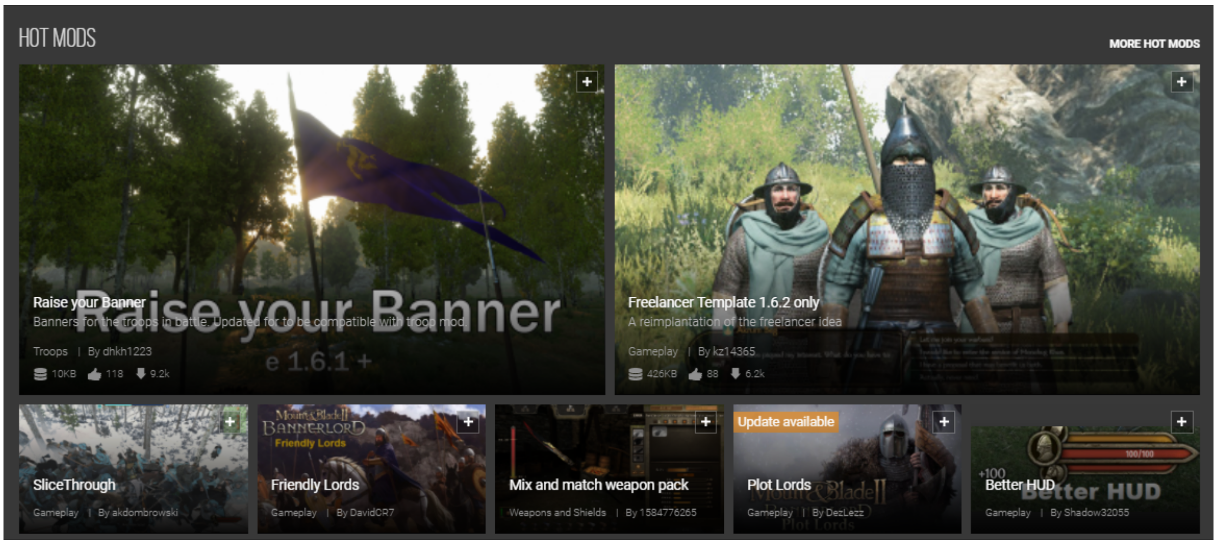

HOT MODS
MORE HOT MODS
Raise your Banner
Banners for the troops in battle. Updated for to be compatible with troop mod
Troops | By dhkh1223
10KB 118 9.2k
Freelancer Template 1.6.2 only
A reimplantation of the freelancer idea
Gameplay | By kz14365
426KB 88 6.2k
SliceThrough
Gameplay | By akdombrowski
Friendly Lords
Gameplay | By DavidCR7
Mix and match weapon pack
Weapons and Shields | By 1584776265
Update available
Plot Lords
Gameplay | By DezLezz
Better HUD
Gameplay | By Shadow32055

Figure 6.1: Friendly Lords on Hot Mods category

Meanwhile, in general, the community's feedback is very positive about the mod and about CiF modifications on the game. Some examples are that looks promising, they want an active development and are thankful to help with bring a new life to the game.

People still giving suggestions and constructive feedback to implement on mod, meanwhile a few groups of players showed more interest. A player contacted the researcher to collaborate on the development, by adding new dialogs to conversations between NPCs and another added a dialog's translation to Turkish, that is the same language as the game's studio. Also, a third player that made a YouTube video talking about some mods and Friendly Lords was one of them. Also, he is testing the voice acting feature simultaneously with the dialogs, where the

Player is able to listen and to read what the characters are talking about, improving the player's experience.

Some relevant comments from players below:

- *"Cool idea, much needed in this game that, aside from combat and economics, is all about being friends and enemies with npcs."*
- *"Lol I saw that mod on the nexus yesterday and literally was thinking dang it sounds amazing"*
- *"I believe you struck gold with this idea and I think together we can make it one of the best (if not THE best) immersion mods out there."*
- *"yeah i think you and the other modder's are the best hope of putting life into this game"*
- *"I can't wait to see the mod developing into something more, it is already gives a lot of immersion while it takes an eternity for taleworld to do their parts. i'd wish to see some kind of chatter (sims like hmming) between soldiers during battle."*
- *"First I went into a tavern and I was pleasantly surprise when I see random people walking around talking to other people."*

Endorsements	35
Unique DLs	1,378
Total DLs	1,956
Total Views	18,020

Table 6.1: FriendlyLords Stats on NexusMods

On ModDB, it is only possible to update and download mods, being unable to interact with the community. With a total of 189 downloads and 3,765 visits.

All the previous data mentioned before was updated on 24th September.

6.2.2 Quantitative data

We had a total of 33 participants that experimented the game and filled out the questionnaire. Of these participants, 78,8% were male, 12,1% were female and 9,1% prefer not to say. Regarding the age, most were in the ages of 19 to 25 years (48,5%), and 26 to 39 years old (36,4%). Only one participant has not played RPGs before. Most said that they play an average of 6 to 12 hours a week, and, an average of 13 to 20 hours a week (each one with 30,3%), as shown in Figure 6.2.

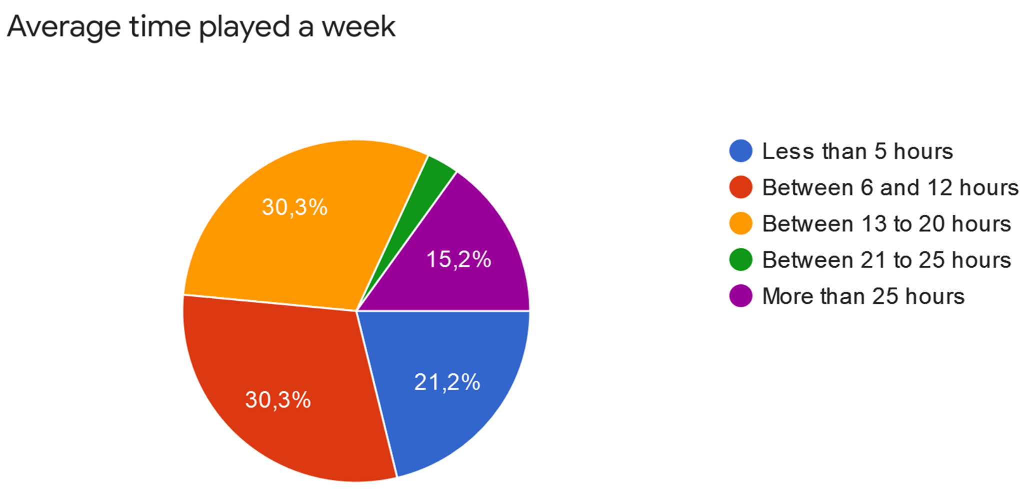


Figure 6.2: Average Time Played

We started by conducting a *Cronbach's Alpha* test for the questions regarding the Believability and Social Presence scales, to determine if we could aggregate the individual questions into a single variable for each scale. It was not necessary to perform this for Enjoyment since we had only one question. The Cronbach's Alpha values for each metric and condition (Condition A is the version without CIF-BL and condition B is the version with CIF-BL) can be consulted in Figure 6.2.

	Condition	
	A	B
Believability	.908	.860
Social Presence	.921	.850

Table 6.2: Cronbach's Alpha results

The cronbach's alpha coefficient indicates that the Believability and Social Presence items have good and excellent internal consistency, therefore we can aggregate the different questions into two main variables: Believability and Social Presence.

Then, we have applied *Shapiro-Wilk Normality Tests* for the aggregated means to determine if the data was parametric. The results are displayed in Table 6.3. The *p-values* for Believability Mean and Social Presence Mean were greater than 0.05, so the data is parametric. For the enjoyment variable (single question) the data was found to not be parametric, hence a non-parametric test will have to be used.

For these tests to reveal statistical significance the resulting p-value must be less than 0.05.

	Condition	
	A	B
Believability	.694	.375
Social Presence	.927	.501
Enjoyment	<.001	<.001

Table 6.3: Normality Tests results

For the successful tests we will also calculate 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 a strong effect.

6.2.2.A Believability

Figure 6.3 shows the result for the believability scale for both versions of the game. We conducted a paired t-test, which indicates that there are statistically significant differences between the two versions of the game with a moderate effect size (p<0.001, d=-0.685). This means that participants perceived the NPCs in the Friendly Lords to be more believable than the original version of the game. The Believability improved from 3.5714 to 4.8571, an increase of 35%.

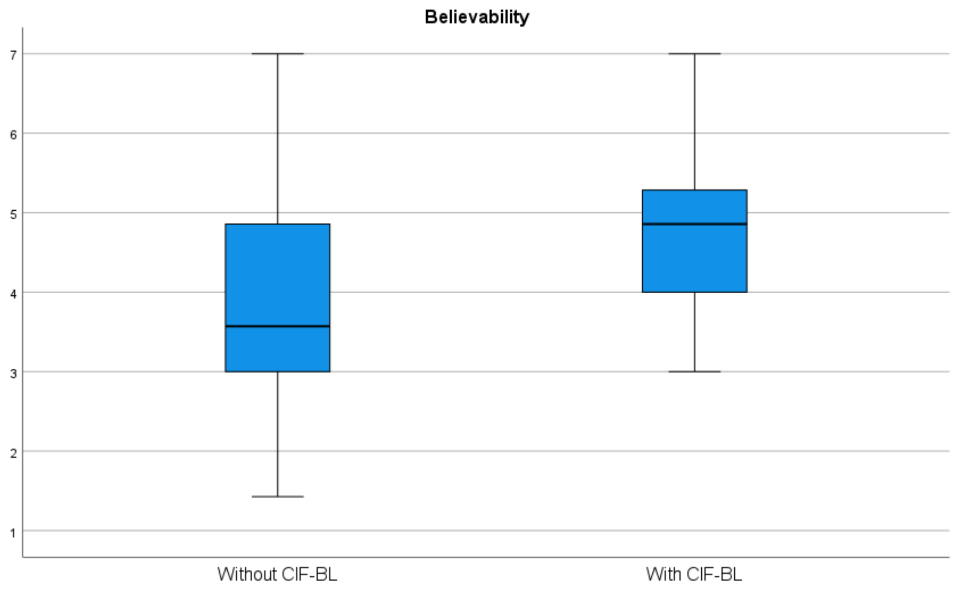


Figure 6.3: Box Plot for Believability

6.2.2.B Social Presence

For the Social Presence chart, in Figure 6.4, the first impression is that the NPCs with CIF-BL seems to have a higher social presence than the NPCs from the original version of the game. A paired t-test confirms this hypothesis, showing a statistically significant difference ($p<0.001$, $d=-0.683$) with moderate effect size. The Social Presence improved from 3.6667 to 4.8889, an increase of 33%.

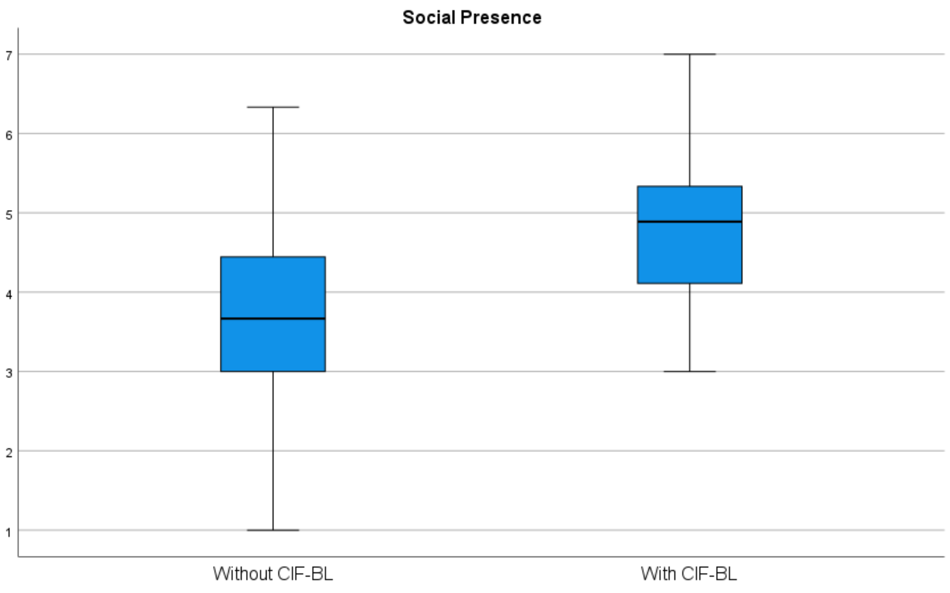


Figure 6.4: Box Plot for Social Presence

6.2.2.C Enjoyment

Regarding the Enjoyment variable, Figure 6.5 does not appears to show a clear difference between the two versions. Only a small adjustment on the lowest boundary for the question "I enjoyed playing the game". Given that the data was not parametric, we applied a Wicoxon Signed Rank test which indeed confirmed that there were no statistically significant differences between the two versions ($p=0.668$).

6.2.2.D In-Game Interactions

The CIF-BL Manager has a functionality to save on one anonymous file, named *user.json* located on the mod's folder. The file contains the information about how many days it passed

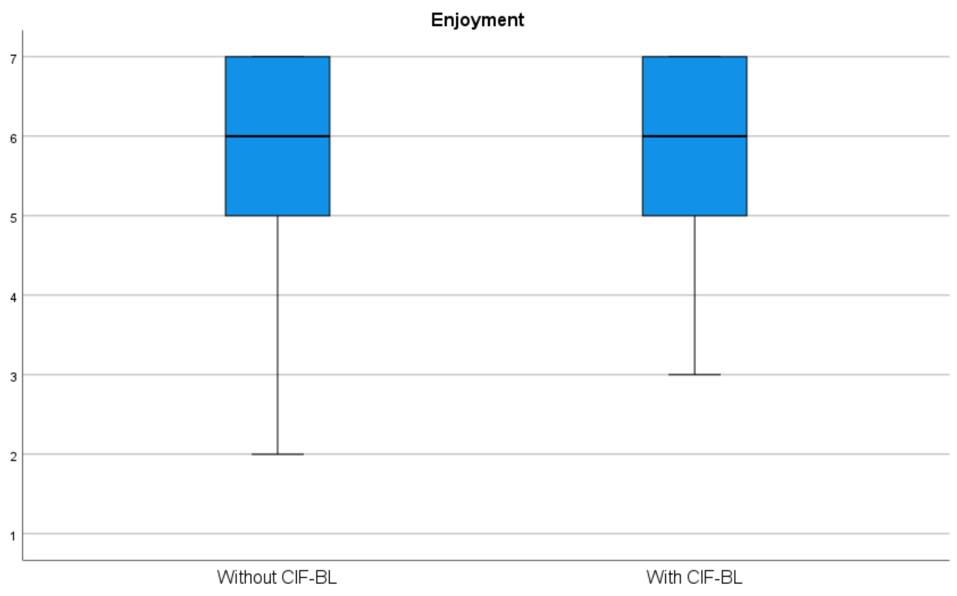


Figure 6.5: Box Plot for Enjoyment

on the game, the social exchanges performed by Player and by the NPCs, with their respective intentions. The file was uploaded a file to a server database.

When analysing the user files, we can see a high number of performed *Compliment SE* and *Flirt SE*, *Positive* and *Romantic* Intentions, as shown in Figure 6.6. Which will increase the Player's relation with NPCs.

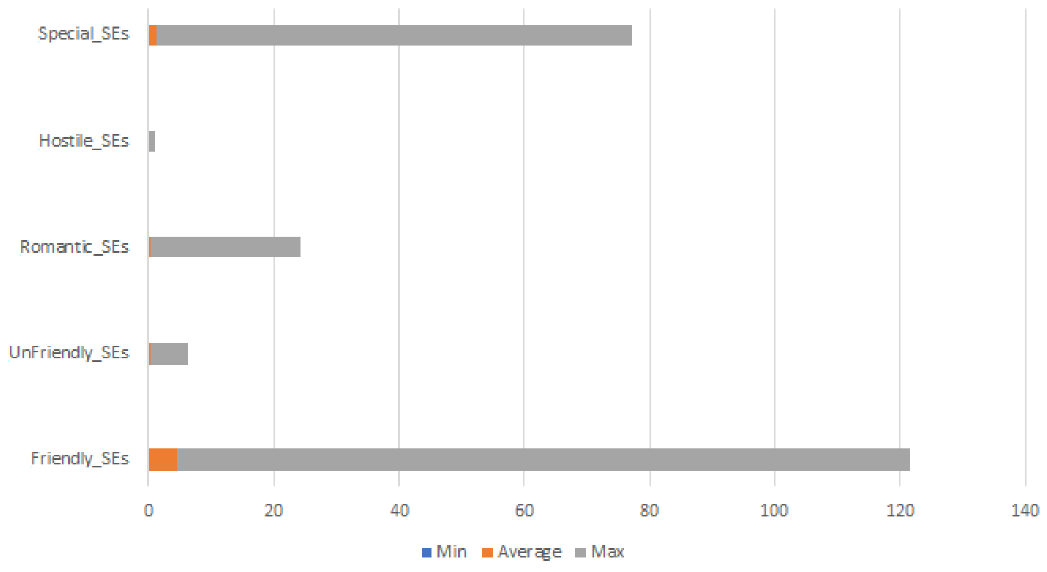


Figure 6.6: Intention's Frequency by Player

There is an advantage in the game itself in having positive relationships because it increases the relationship with NPCs, being a reason with high probability. There is no benefit to having negative relationships in the game (hence there are no negative social exchanges with NPCs).

	Player -> NPC	NPC -> Player	NPC -> NPC	Total SEs
Average	6.25	4.86	22.47	33.58
Max	216	188	359	589
Total	3596	2796	12918	19310

Table 6.4: Interactions Overview

On Table 6.4, it is possible to see the information about how many times Player had interacted with NPC, a NPC interacted with Player and with other NPC, and, the total of Social Exchanges.

The previous data was updated on 19th September.

6.3 Discussion

Despite the limitations of the study - given the fact that it was conducted in an online uncontrolled setting - the results obtained are quite good. The proposed and implemented CIF-BL model was indeed able to improve the social interaction of NPCs in the game Bannerlord, making the NPCs to be perceived as more believable and with higher social presence. We believe that this will lead to a more engaging, diverse experience.

Unfortunately, we could not find a direct impact of the CIF-BL model in participant's enjoyment of the game. We argue that this can possibly be explained by the fact that participants already knew and liked the game. That is, they already play the game because they like it, so, they are naturally biased. Looking at the chart from 6.5 we can see that participants rated both version with very high enjoyment (median 6), and in such situations it is very hard to find an effect.

Although the questionnaires did not find a difference in enjoyment, the qualitative feedback obtained directly from participants lead us to conclude that in general players from the community are enthusiastic about this type of changes (improving NPCs social behaviour) and that they believe that this type of modifications are really important to improve the game.

7 Conclusion

Contents

7.1 Conclusion

In this thesis, we have looked at a major problem in modern video game development: believability and quality of social interactions of Non-Playable Characters (NPCs).

We created the CiF-BL architecture that is the result of adapting and implementing CiF to Bannerlord's game with some innovations such as a centralized controller system and its effect on the NPCs' decision making cycle.

We were successful in creating CiF-BL model and implementing it as a mod. The mod uses 11 different Social Exchanges, 10 different traits, and, 5 different Status, designed to allow users to use all of the CiF-BL architecture.

The mod is called "Friendly Lords" and was released online in the 18th of August both in a popular mod websites: "Nexus Mods" and "ModDB". In ModDB it had a total of 189 downloads and 3,765 visitors. In Nexus Mods it had 35 endorsements, it has been played by over 1300 different players, along with comments supporting the mod and asking us to improve it with more social exchanges and dialogs (and for voice support). The mod page has also received over 18,020 different visitors. It also reached the status of "Trending" , "Most Downloaded (30 days)" and "Hot Mods" because of its popularity.

We conducted an online evaluation with 33 participants in order to test the success of the implementation of CiF-BL model. The results were very positive, users clearly preferred the CiF-BL based NPCs and considered our iteration of NPCs more believable and with higher social presence.

Additionally, the mod's reception and popularity lead us to conclude that this project was successful in achieving its goals. Using an Artificial Intelligence Social Architecture we were able to create more believable characters and to improve the NPCs social behaviour

7.1.1 Future Work

In the future, we can improve the work that has already been done it by adding more traits, status, different types of Social Exchanges, more events and trigger rules and improve the existent, to give to players a more immersive experience. Also, more options to react with more and less intensity, to use the familiar relationships on the game, and to add more dialogs' options/variety. Additionally, a controlled evaluation with in-person participation would allow us to better understand the impact of the developed mod in user experience and enjoyment.

It can also be implemented the voice acting feature, to user be able to listen the characters, and, a conversational AI to generate the NPCs' dialogues on runtime, instead of being preloaded.

Bibliography

[1] S. Rank, “Affective acting: An appraisal-based architecture for agents as actors,” *OGAI Journal (Oesterreichische Gesellschaft fuer Artificial Intelligence)*, vol. 26, no. 3, pp. 7–10, 2007.

[2] J. Dias, S. Mascarenhas, and A. Paiva, “FAtiMA modular: Towards an agent architecture with a generic appraisal framework,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8750, no. December, pp. 44–56, 2014.

[3] L. S. Zaremba and W. H. Smoleński, *FLAME_Fuzzy_logic_adaptive_model_of_emotions*, 2000, vol. 97, no. 1-4.

[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, 2014.

[5] B. Samuel, A. A. Reed, P. Maddaloni, M. Mateas, and N. Wardrip-Fruin, “The Ensemble Engine: Next-Generation Social Physics,” pp. 22–25, 2015, http://www.fdg2015.org/papers/fdg2015_paper_07.pdf [Last accessed on 25 Nov 2021].

[6] C. Becker-asano, “Wasabi : Affect simulation for agents with believable interactivity dissertation zur erlangung des grades eines at bielefeld,” 2008, https://books.google.pt/books?id=8ABvlwHBCQIC&printsec=frontcover&dq=WASABI+:+Affect+Simulation+for+Agents+with+Believable+Interactivity&hl=en&sa=X#v=onepage&q=WASABI%20%3A%20Affect%20Simulation%20for%20Agents%20with%20Believable%20Interactivity [Last accessed on 25 Nov 2021].

[7] S. C. Marsella, D. V. Pynadath, and S. J. Read, “Psychsim: Agent-based modeling of social interactions and influence,” *Proceedings of the international conference on cognitive modeling*, pp. 243–248, 2004.

[8] P. B. Bernt Meerbeek, Jettie Hoonhout and A. van Breemen, "icat - a friendly robot that helps children and grown-ups," https://ercim-news.ercim.eu/en67/special-theme-embedded-intelligence/icat-a-friendly-robot-that-helps-children-and-grown-ups [Last accessed on 25 Nov 2021].

[9] M. Vala, P. Sequeira, A. Paiva, and R. Aylett, "FearNot! demo: A virtual environment with synthetic characters to help bullying," *Proceedings of the International Conference on Autonomous Agents*, no. May 2014, pp. 1381–1382, 2007.

[10] M. Y. Lim, J. Dias, R. Aylett, and A. Paiva, "Creating adaptive affective autonomous NPCs," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 2, pp. 287–311, 2012.

[11] PlayablStudios, "Façade," https://www.playablstudios.com/facade [Last accessed on 25 Nov 2021].

[12] Maxis, "The sims 3," https://www.thesims3.com/ [Last accessed on 25 Nov 2021].

[13] R. Evans and E. Short, "Versu - A simulationist storytelling system," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 2, pp. 113–130, 2014.

[14] J. McCoy, M. Treanor, B. Samuel, and A. Reed, "Prom week: Designing past the game/story dilemma," pp. 94–101, 2013, https://www.aaronareed.net/papers/Prom%20Week%20FDG%202013.pdf [Last accessed on 25 Nov 2021].

[15] A. Sullivan, A. Grow, M. Mateas, and N. Wardrip-Fruin, "The design of mismanor: Creating a playable quest-based story game," *Foundations of Digital Games 2012, FDG 2012 - Conference Program*, no. April 2016, pp. 180–187, 2012.

[16] M. Guimaraes, P. Santos, and A. Jhala, "CiF-CK: An architecture for social NPCS in commercial games," *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, pp. 126–133, 2017.

[17] J. Mccoy, M. Mateas, and N. Wardrip-fruin, "Comme il Faut : A System for Simulating Social Games Between Autonomous Characters," 2009, http://escholarship.org/uc/item/6x5933cw.pdf [Last accessed on 25 Nov 2021].

[18] J. O. Frank Thomas, *The Illusion of Life Disney Animation*, 2014.

[19] P. Hingston, “Believable bots: Can computers play like people?” pp. 1–318, 2013, https://www.researchgate.net/publication/266895579_Assessing_Believability [Last accessed on 25 Nov 2021].

[20] P. A. Dias J., “Feeling and reasoning: a computational model for emotional agents.” 2005, https://pdfs.semanticscholar.org/1b11/4e2086b356b9a29ec93d8f4c72a1fb5bca73.pdf [Last accessed on 25 Nov 2021].

[21] J. Dias, “FearNot !: Creating Emotional Autonomous Synthetic Characters for Emphatic Interactions,” 2005.

[22] W. Scacchi, “Introduction Mods, modders, modding, and the mod scene Discussion Conclusions,” 2010, http://firstmonday.org/article/view/2965/2526 [Last accessed on 25 Nov 2021].

[23] Z. Kowalczuk and M. Czubenko, “Computational Approaches to Modeling Artificial Emotion – An Overview of the Proposed Solutions,” 2016, https://www.frontiersin.org/article/10.3389/frobt.2016.00021 [Last accessed on 25 Nov 2021].

[24] S. H. Rodrigues, S. F. Mascarenhas, J. Dias, and A. Paiva, “"i can feel it too!": Emergent empathic reactions between synthetic characters,” *Proceedings - 2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, ACII 2009*, no. August 2019, 2009.

[25] C. Morais, “From caveman to gentleman implementing an artificial intelligence social architecture in conan exiles,” no. October, 2018.

[26] J. McCoy, M. Treanor, B. Samuel, B. Tearse, M. Mateas, and N. Wardrip-Fruin, “Authoring Game-based Interactive Narrative using Social Games and Comme il Faut,” 2010, https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=743033D3D4A9BE0C5720D3F51DDA9EE8?doi=10.1.1.592.9203&rep=rep1&type=pdf [Last accessed on 25 Nov 2021].

[27] J. McCoy, M. Treanor, B. Samuel, N. Wardrip, M. Mateas, and M. M. Fruin, “Comme il Faut: A system for authoring playable social models,” *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2011*, pp. 158–163, 2011.

[28] R. Today, “icat,” https://www.roboticstoday.com/robots/icat-description [Last accessed on 25 Nov 2021].

[29] A. Rao and M. Georgeff, “BDI Agents: From Theory to Practice,” *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 1995.

[30] B. G. Studios, “The elder scrolls v: Skyrim,” https://elderscrolls.bethesda.net/en/skyrim/ [Last accessed on 25 Nov 2021].

[31] Funcom, “Conan exiles,” https://www.conanexiles.com/ [Last accessed on 25 Nov 2021].

[32] R. Aylett, A. Paiva, N. Vannini, S. Enz, E. Andre, and L. Hall, “But that was in another country: Agents and intercultural empathy,” *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 1, no. May 2014, pp. 475–482, 2009.

[33] M. Kriegel, M. Yii, A. Nazir, A. Cawsey, S. Enz, P. Rizzo, and L. Hall, “ORIENT : An Inter-Cultural Role-Play Game,” pp. 1–4, 1984.

[34] M. Mateas and A. Stern, “Façade: An experiment in building a fully-realized interactive drama,” *Game Developers Conference*, vol. 2, 2003.

[35] C. S. 4, “The sims 4 emotions, "emotion system mechanics, moods, and how to get sims feeling each",” https://www.carls-sims-4-guide.com/emotions/ [Last accessed on 25 Nov 2021].

[36] E. Short, “About versu,” https://versu.com/ [Last accessed on 25 Nov 2021].

[37] R. P. Evans and E. Short, “The AI Architecture of Versu,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 2, pp. 1–37, 2014.

[38] T. Senior, “Mount & blade 2: Bannerlord announced with tiny teaser trailer,” https://www.pcgamer.com/mount-blade-2-bannerlord-announced/ [Last accessed on 25 Nov 2021.

[39] I. Millington, *AI for Games*. Third Edition, CRC Press, 2020.

[40] N. Parde and R. D. Nielsen, *User Perceptions of a Conversational Robot Interface*. Association for Computing Machinery, 2019, vol. 1, no. 1.

[41] J. A. Mccoy, “All the world’s a Stage: A Playable Model Of Social Interaction Inspired by Dramaturgical Analysis,” 2012.

[42] P. F. Gomes, “MEEMOs: Believable Agents with Episodic Memory Retrieval,” p. 87, 2010.

[43] C. N. Gunawardena and F. J. Zittle, “Social presence as a predictor of satisfaction within a computer-mediated conferencing environment,” pp. 8–26, 1997, https://doi.org/10.1080/08923649709526970 [Last accessed on 25 Nov 2021.

[44] J. Short, E. Williams, and B. Christie, “The social psychology of telecommunications,” 1976, http://lib.ugent.be/catalog/rug01:000424961 [Last accessed on 25 Nov 2021].

[45] C. Harms and F. Biocca, “Internal Consistency and Reliability of the Networked Minds Measure of Social Presence,” *Seventh Annual International Workshop: Presence 2004*, pp. 246–251, 2004.

[46] R. Ryan, “Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory.” *Journal of Personality and Social Psychology*, vol. 43, pp. 450–461, 1982.

A

Tables and Figures

Term	Function
Intention	Enum_CIF_Intention
SocialExchange	Enum_CIF_SocialExchange
Influence Rules	General rules applied when calculating the volition
Volition Function	Function that calculates the initiator or receiver volition
Initiator	Agent
Receiver	Agent
ThirdAgent	Agent
Animation	Returns an Animation
Outcome	Integer

Table A.1: CiF-BL Social Exchange

Social Exchange	**Description**
Compliment	The initiator says something pleasant about the receiver. Increases relationship. Only available if they are not dating.
GiveGift	The initiator offers an item to the receiver. Increases relation. Only available if the initiator has an item to offer.
Admiration	The initiator will congratulate the receiver for an event that occurred. Increases relationship. Only available from NPCs to Player and when Player's Clan level up.
Jealous	The initiator says something unpleasant about the receiver. Decreases relationship. Only available if it's not dating with the receiver.
FriendSabotage	The initiator says something unfriendly about a third agent. Available when the initiator has negative relationships.
Flirt	The initiator tries to charm the receiver. Increases Relationship. Only available when the initiator is dating with the receiver.
Bully	The initiator says something unpleasant about the receiver. Decreases the relationship. Only available if it's dating with the receiver.
RomanticSabotage	The initiator says something unfriendly for the receiver. Available only when someone has romantic interactions with initiator's partner.
AskOut	The initiator asks the receiver to be a couple. If the receiver accepts the Dating relationship is set for the both of them. Only available if the initiator is not dating the receiver.
Break	The initiator will end their Dating relationship with the receiver. This Social Exchange option is only available to couples.
HaveAChild	The initiator asks about have a family with receiver. Only available when the couple's relation is high.

Table A.2: Short description of the Social Exchanges

Element Name	**Implementation**
AgentReference	Agent
CiF Name	String
CiF ID	Integer
cultureCode	CultureCode
EnoughRest	Boolean
Busy	Boolean
IsInitiator	Boolean
Target	Agent
ID Target	Integer
ThirdAgent	Agent
ID ThirdAgent	Integer
FullMessage	Array of Strings
Message	String
SE_Accepted	Boolean
CIF_NPCs	List of AgentReferences
Countdown	Float
TopSE	List of Social Exchanges
Social Network Beliefs	List of Object_SNBelief
Social Exchange Memory	List of Object_MemorySE
ItemList	List of Strings
StatusList	List of Strings
SocialExchange	CiF_SocialExchange_Enum

Table A.3: CiF-BL Character

Element Name	**Implementation**
Friendly	The volition for Positive SEs is increased. The volition for Negative SEs is decreased.
Hostile	The volition for Negative SEs is increased. The volition for Positive SEs is decreased.
Charming	The volition for Romantic SEs is increased.
UnCharming	The volition for Romantic SEs is decreased.
Shy	The volition for SEs in general is decreased. The Social Exchange countdown for this character will also be increased.
Brave	The volition for SEs in general is increased. The Social Exchange countdown for this character will also be decreased.
Calm	The volition for Positive SEs is increased. The volition for Negative SEs is decreased.
Aggressive	The volition for Negative SEs is increased. The volition for Positive SEs is decreased.
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.

Table A.4: Short Description of the Traits

Term	Function
None	There are no conditions. The dialog will be displayed.
FriendlySE_condition	Check if Player has *Friend* relation to perform SEs with Positive Intention
UnFriendlySE_condition	Check if Player has *Friend* relation to perform SEs with Negative Intention
AskOut_condition	Check if Player is available to perform AskOut SE
Romantic_condition	Check if Player has *Dating* relation to perform SEs with Romantic Intention
Hostile_condition	Check if Player has *Dating* relation to perform SEs with Hostile Intention
Break_condition	Check if Player is available to perform Break SE
RomanticAdvanced_condition	Check if Player has *Dating* relation with high value to perform more SEs with Romantic Intention
NPCReactsToAskOut_condition	Check how the character will react (true or false) to AskOut SE
PlayerOfferGift_condition	Check if Player has any item to perform GiveGift SE with companion
CompanionOfferGift_condition	Check if companion has any item to perform GiveGift SE

Table A.5: Conditions implemented on CiF-BL Dialogs

Term	Function
None	There are no consequences
Increase_Relation	Increases the relation’s value
Decrease_Relation	Decreases the relation’s value
Start_Dating_consequence	Characters start Dating relationship
DoBreak_consequence	Characters will be no longer Dating
PlayerGiveItem_consequence	Player will lose an item
RomanticAdvanced_consequence	Increases the relation’s value

Table A.6: Consequences implemented on CiF-BL Dialogs

B

Questionnaire

	Strongly Disagree	Disagree	Somewhat Disagree	Neither Agree nor Disagree	Somewhat Agree	Agree	Strongly Agree
The NPCs' behavior was predictable.							
I liked the NPCs.							
The NPCs' behavior drew my attention.							
The NPCs perceived the world around them.							
The NPCs' behavior was interesting.							
The NPCs were intelligent.							
It was easy to understand what the NPCs were thinking about.							
The NPCs had personality.							
I enjoyed playing the game.							
The NPCs noticed my character.							
The NPCs caught my attention.							
The NPCs found it easy to understand my character.							
I could tell how NPCs felt.							
The NPCs could tell how my character felt.							
The NPCs attitudes influenced how I felt.							
My character's attitude influenced how the NPCs felt.							
My behavior was often in direct response to the NPCs behavior.							
The NPCs behavior was often in direct response to my character's behavior.							

Table B.1: Metrics Questionnaire