A System to Simulate Culture in Role Playing Games

INF2102 - Projeto Final de Programação

Name: Luís Fernando Teixeira Bicalho

Registration: 1921367

July 2021

Advisor:

Prof. Bruno Feijó

1 Abstract

Video game players are often seeking for deeper experiences with game characters. Most of the time, they want to relate to each game character in a different way. More specifically, they like to have non-player characters (also known as NPCs) reacting differently and in a more realistic way, depending on the action the player takes. However, believable game characters should react emotionally, reveal personality traits, and incorporate the characteristics of the region they live in the game world. This is why we propose a different approach to the problem of modeling game character behaviors. In this new approach, these behaviors are not only based on the emotions and personality traits of each character, but also based on the culture in which each NPC is inserted. In the present project, we develop a system to simulate cultural behavior of game characters, which is closely aligned with well-known emotion and personality models. Our goal is to simulate the cultural behavior based on six different dimensions: wealth, rationality, politeness, collectivism, time and dignity. Furthermore, we propose the integration of these dimensions with the concepts of trust/confidence level, prejudice, personality, and emotion. We tested the model by creating an experimental Role Playing Game (also known as RPG), called *Future Falls*, which can be played through its builds (can be accessed in the following link: https://github.com/Kirink212/future-falls). This helped us create more believable game characters and make the game experience richer and more diverse.

Key Words: NPCs behavior; Culture model; Game AI; Cultural behavior; Emotion and Personality Models; Proxemics;

2 Introduction

The game industry is increasingly growing year by year [1], moving lots of money and calling investors' attention for this market [5]. Also, Brazil is the 13^{th} country in game consumption and the bigest in Latin America [7], being one of the best countries to invest in this area. Because of that, production is continuously growing, making the competition increase but also increasing market saturation with some game genres [8]. In this context, to make a new stand out in the middle of so many, game designers must bring a different and unique experience for the player [9].

Therefore, the industry always thought of ways to make the gameplay different from its competitors, also trying to expand game's lifetime, increasing game's replayability (i.e. the video game's potential for continued play value after its first completion). Among all the strategies that allow this variety to be applied, the most known is to give freedom to the player itself through the open-world experience [10], showing the world and tools for the player to create his own game. In this context, one of the earliest successful example was the game Elite (1984), which started the *space sim* genre [11]. This sense of variety is even bigger when we talk about Procedural Content Generation, that makes game like *Minecraft*[26] and *No Man's Sky*[20] stand out as examples of millions of interactions during gameplay, causing it to take over the game industry[6].

However, the above-mentioned strategies usually focus on map creation, not giving attention to the way NPCs react to player actions throughout the game. To start solving this issue, we adopt the strategy in which the NPCs actions are based on behavior models that can relate to emotions and personality in lots of different ways. In this project proposal, we will focus on two widely known models to help simulating behavior: Plutchik's Wheel of Emotions[28], to help us simulate emotions; OCEAN personality model[24] (also known as the Big Five model), to help us simulate personality traits.

Lots of games and projects use these models to help simulating behavior, like in [14] and [33], but there are few of them that try to simulate a behavior based on cultural background (i.e. on cultural elements), and none of them apply and test them in the video game context. Thus we will present, in this proposal, a model to create game characters that are inserted in a specific culture. Moreover, we shall make them behave accordingly to the player's actions.

Nevertheless, we will not limit ourselves to create NPCs from specific cultures, but make their own actions depend on their culture, personality, current emotion and two more individual factors: prejudice, which is related to discrimination towards others; confidence or trust, being how much the NPC trusts the main player. This last factor is already used in some games, like *Binary Domain* [31], in which the way the player protects his/her allies during battle, the things he/she says and do influence the confidence level of each character differently, and bringing future consequences to the narrative [2].

Also, to test the model above mentioned, we pretend to develop an experimental RPG, called *Future Falls*. We focused on RPGs, because games from this genre are constantly being created and are highly accepted by the great public[3]. In the proposed game, we will play as a character from the Human race (here, race is a synonym for culture) who tries to end human slavery and make peace with other cultures. To do that, he/she will need to interact with different races, in different regions, and gain their trust by doing specific missions for each different NPC, or interacting with them buying or selling items, or performing other actions.

This report is organized as follows. The section *Related Works* presents the works that needed to be studied or read, with special focus on behavior models. *Game System* mainly presents how the game requirements were defined, what are the possible inputs (controls and mechanics) and outputs (NPCs behaviors and game screens/interfaces); *Emotion, Personality and Culture Classes/Models* presents how these three models were defined in game, how they were implemented, and how the system interpret their values in order to cause a reaction of the NPC towards a Player's action; *System Modeling, Requirements, and Implementation* presenting the system models (UML classes), functional and non-functional requirements, and how models and some methods were implemented; *Conclusion* and *Maintenance* focus on summarizing the contributions of this project and the future works that can be developed inspired by our results.

3 Related Work

The project that directly influenced the present proposal was the work by Baffa et al (2017) [14], *Dealing with the emotions of Non Player Characters.* Their work presents an experimental game in which emotion and personality models (namely, Plutchik's Wheel of Emotions and OCEAN models) influence the way NPCs behave towards player actions. That research work has shown good results compared to other approaches, while its implementation is kept simpler and concise.

The other research work that inspired our project proposal, contributing for the concept of culture model, is the paper by Böloni et al. (2018) [16]: *Towards a computational model of social norms*, in which a model for simulating social interaction is explained. This model can be used not just in an interaction between a virtual agent and a human user, but also to predict human behavior. It maps values that certain cultures find desirable, like wealth, time, dignity, politeness and generosity.

The five culture traits presented in [16] are based on the Geert Hoftede's theory, called *Cultural Dimension Theory.* The most recent revision of this theory considers six quantitative cultural dimensions[22]:

1. power distance, the acceptance of unequal distribution of power;
2. individualism versus collectivism;
3. uncertainty avoidance;
4. masculinity versus femininity, a metric measuring the balance between assertiveness and competitiveness versus a focus on cooperation, human relations and quality of life;
5. long term versus short term orientation;
6. indulgence versus self-restraint;

Furthermore, Hofstede's analysis shows us that even if two cultures define the same set of social metrics, they might weight these differently in practical behavior. Our model also uses six dimensions, but adapting the names and concepts to the game context, using the same concepts defined by Böloni [16].

4 Game System

This section explains how the RPG *Future Falls* works and how the emotion, personality and culture models fit into the gameplay. The game was developed using the *Unity* game engine, as it is one of the most used in market [4]. Moreover, each one of the following sections is based on *Game Design Document* topics, as seen on the book *Level Up! The Guide to Great Video Game Design* from Scott Rogers [30]. Therefore, in this section we present: Game Mechanics; Game Controls (Inputs and Outputs); Game Visualization; and Game Screens.

4.1 Game Mechanics/Possible Player Interactions

We decided to develop *Future Falls* mechanics following the base Role Playing Game genre features: *experience points*, that exist to help each character to level up and upgrade his abilities; *dungeons*, that are places in which the player fight enemies and level up faster, also obtaining new items; *narrative choices*, that allow the player to build its own story and interact different ways with NPCs.

Bringing RPG features to the *Future Fall's* context, makes it possible for the player to have its own experience level, which can influence the following general items: the clothes he/she can wear, that is, the armor level; the weapons he/she is able to use; the abilities and the energy he/she has.

As the game is based on players actions towards NPCs, he/she has the freedom to choose the best way to make an approach or talk to them, using his previous relations to guide him through this adventure. However, not all attitudes are considered positive for every NPC, depending on their cultural dimensions' values.

1. Walk in the four directions: If the player surpasses no social space from any NPC, he/she won't have any type of reaction;

2. Shoot with a gun in the mouse/analog direction: If the player tries to kill someone, the NPC which is near (public space) will react to it in a positive or negative way, depending on his collectivism value. Moreover, if the gun hits the own NPC, he/she only reacts negatively;

3. Push/Touch a NPC: Positive or negative, depending on NPC's dignity level;

Mental State	Trust Level Influence
Anger	Influences negatively (-1)
Fear	Influences negatively (-1)
Trust	Influences positively (+1)
Disgust	Influences negatively (-1)
Joy	Influences positively (+1)
Sadness	Influences negatively (-1)
Surprise	Influences positively (+1)
Anticipation	Influences negatively (-1)

Table 1: Mental states and its related factors

4. Talk to someone (politely or not): ;

5. Give/Steal item from NPC: Always influence a negative reaction, varying its intensity depending on NPC's wealth value;

6. Give/Steal money from NPC: Influence a positive or negative reaction, depending on NPC's dignity value.

7. Approach the NPC (proxemics): Influence a positive or negative reaction, depending on NPC's dignity level.

Each enemy has a trust level (tLvl), which is influenced by any of the above cited player actions. The positive or negative reaction we talked about means that each action can make the NPCs trust level lower or increase, depending on the positive or negative reaction. Thus, as seen in table 1, each NPC mental state (resulted from the emotional reaction to player's behavior) influences positively (+1) or negatively (-1) the current NPC's trust level (mentalFactor). Checking the equation 1, we can notice that this factor is multiplied by the prejudice level (pLvl), a percentage that determines the total value of this influence. Also, t is defined as the current time. On the list below, we explain each of this levels more specifically.

$$tLvl_{t+1} = tLvl_t + pLvl \times mentalFactor \tag{1}$$

1. Trust Level: It is a float value, from 0 to 1, that represents how trustful the player is to that specific character, always starting equals to 0.5, representing uncertainty.
2. Discrimination Level: It is a float value, from 0 to 1 (percentage), that influences the NPC trust level.

4.1.1 Iteraction Input/Output Example

Let's imagine a situation in which the player tries to sell an item to a NPC, interacting with him/her for the first time. The action of approaching the NPC, depending on his/her personality, may make him keep walking (neutral emotion) or walk in a lower speed, almost stopping (surprise emotion). He/she can also run worried (anticipation emotion), but just if the player has already encountered him/her, which is not the case here, and he/she reacted in a way that decreased his/her trust level towards the player, or his/her discrimination level towards humans affected the situation negatively.

After that, the player can offer an item to him/her. If his/her discrimination level does not influence his/her reaction, he/she can decline it right away or ask the player how much does it cost. If he/she accepts the player's first offer, the interaction ends there, otherwise he/she may decline it or offer a lower price. Each reaction will depend on his/her current emotion and personality.

Besides that, there is also NPC's cultural dimension values that may influence the result of the situation. For instance, if his/her time value is 0.2, this means that he/she will spend just 20% of the time limit (arbitrary value that exists for each different type of interaction, being 3 minutes for a sale operation), which is 36 seconds. This results in the NPC leaving the player when the time spent during the conversation equals the limit time.

4.2 Non Playable Characters Expected Behaviors

When a NPC reacts to a player action, he changes his mental state, a value that represents his current emotional state, i.e. the value based on the most influent emotion. As shown by Table 2, each mental state is related to a NPC behavior.

As our game is a 2D adventure, it was really difficult to find a way to show emotions like sadness or joy in a NPC. Therefore, we created a 3D application inside the game, in

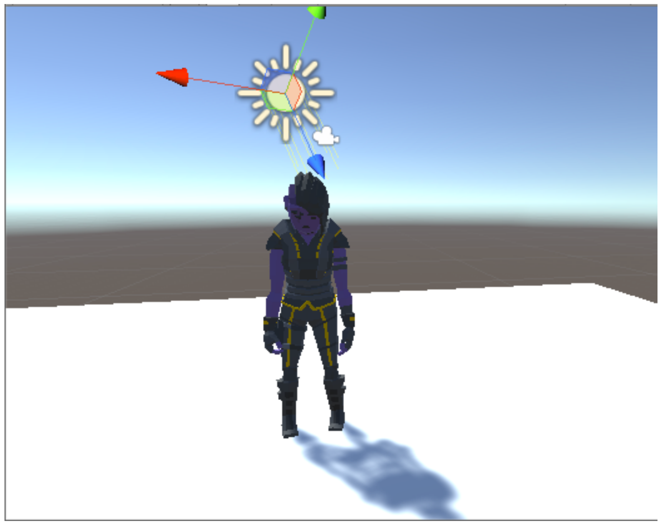

Figure 1: NPC's behavior in *Sadness* mental state

which 3D models would react based on the proposed mental states. For instance, if the current mental state is *Sadness*, the NPC would react like in Figure 1. To change the current mental state, the user must press the correspondent key (shown in 2), making the NPC transit between animations.

4.3 Game Inputs

The interaction between user and game is limited through the controller, that can be a computer keyboard or a joystick. The keyboard keys and the outputs generated are:

1. Key "W": Change player animation to the *walk_up_state*, making him/her move on the positive direction of the Y axis;

2. Key "A": Change player animation to the *walk_left_state*, making him/her move on the negative direction of the X axis;

3. Key "S": Change player animation to the *walk_down_state*, making him/her move on the negative direction of the Y axis;

4. Key "D": Change player animation to the *walk_right_state*, making him/her move on the positive direction of the X axis;

5. Key "J": Change player animation to the *attack_state*, making him/her shoot with the gun currently being held;

Mental State	NPC Behavior in Game	NPC Behavior in 3D Application	Key
Fear	Run Away	Terrified Animation	3
Anger	Shoot in player's direction	Yelling Animation	4
Trust	Follows player for 5 seconds	Thankful Animation	8
Disgust	Run Away	Loser Animation	7
Joy	Follows player for 5 seconds	Happy Animation	1
Sadness	No reaction	Sad Idle Animation	2
Surprise	No reaction	Surprised Animation	5
Anticipation	No reaction	Excited Animation	6

Table 2: Mental state and NPC's behavior

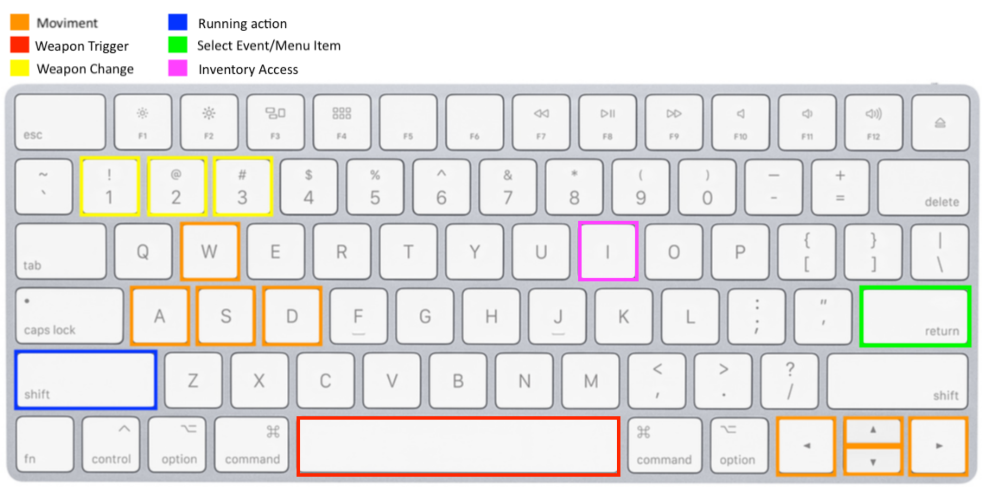

Figure 2: Keyboard Controls

Figure 3: Joystick Controls (follows the same keyboard caption)

6. Key "I": Access your inventory, which contains player's items, weapons and the ones that are currently being used by him/her;

7. Key "SHIFT": Change player animation to the *running_state*, making he/she walk faster to the current direction (the user has to press this button along with any of the walk buttons);

8. Key "ENTER": This key is used in the inventory or the menus to select the current highlighted button or item, that has been chosen by the walk buttons listed above ("W", "A", "S" and "D", to move the menu and inventory selector up, left, down and right, respectively). It is also used to select the current player action related to: items, money and conversation. This only occurs when player and the NPC are "engaged in action", which happens if the player is at least in NPC's public space;

9. Keys "1", "2" e "3": This keys, respectively, are used to choose between the three weapons held by the player.

4.4 Game Visualization

Figure 4: The Legend of Zelda graphics and interface

The game itself is the output of the system, showing the results in image, text and player/NPC interaction format. As we are focusing on the RPG genre specifications, the visuals should follow the market games references, as seen in Figure 4, of the game

Figure 5: *Future Falls* interface

The Legend of Zelda: A Link to the Past[27]. This game's interface, however, is very different from the one implemented in *Future Falls*. As shown by Figure 5, the player has an interface limited to his life (in green color), placed above his character model, and his three mais weapons he/she can choose from during gameplay.

Any other collected or bought weapon is available in the inventory, allowing this three mais to be changed throughout the gameplay. Besides the life bar, the NPCs also have their trust level, represented in blue, above the green bar.

The way the engine display and positions all this elements in the screen is through the *Cartesian Coordinate System*, widely used in mathematics. This system has a X axis, that grows positively in the right direction, and negatively in the left direction, and a Y axis, that grows positively in the up direction, and negatively in the down direction, as shown in Figure 6.

4.5 Game Screens

Future Falls has three main *Unity* scenes: the Main Menu Scene, the Game Scene, and the 3D Application Scene. The Main Menu, as shown by Figure 7, is the first loaded screen, in which the player can choose from three options: go to *Future Falls'* game application, go to the 3D Animation application, or exit the game.

Inside the Game Scene, three Canvas were created: Game Over Canvas, Pause Game Canvas, and Victory Canvas. Visually they are almost the same, changing the title and

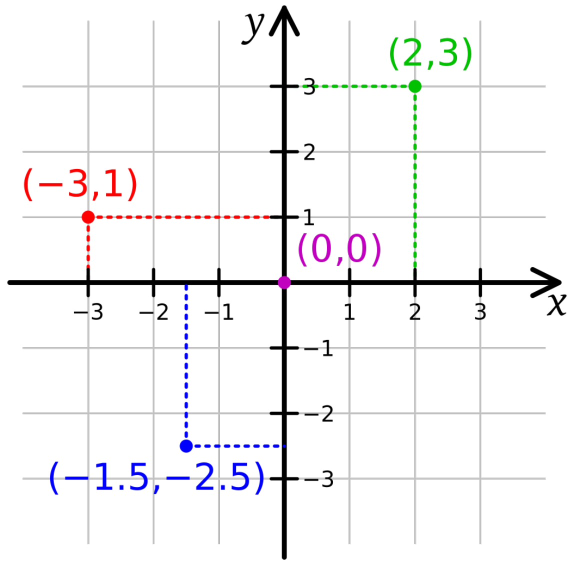

Figure 6: Cartesian plane example

Figure 7: *Future Falls'* main menu screenshot

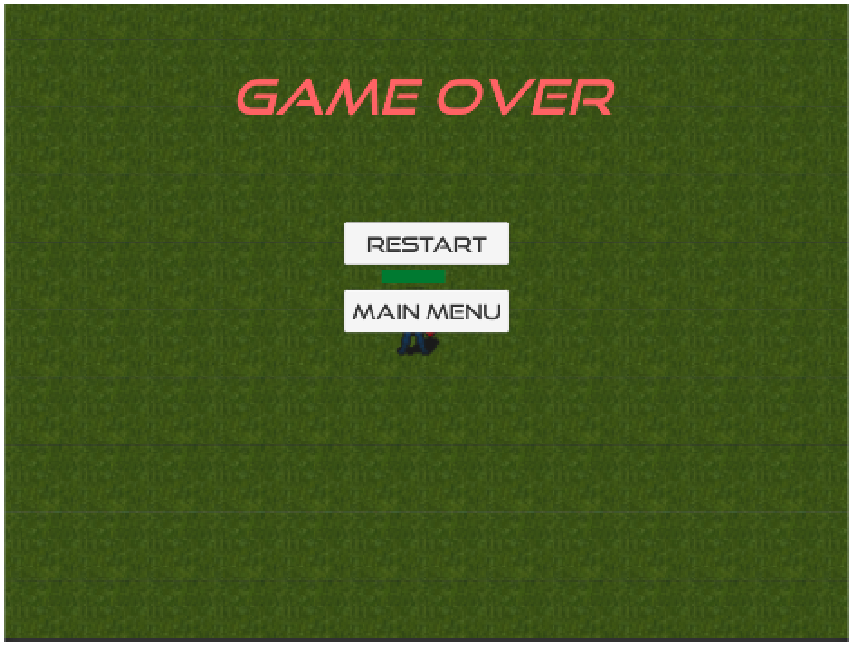

Figure 8: Game Over screenshot

its color. As shown in Figures 8, 9 and 10, the Pause Game Canvas is the only that has the quit button, while all of them have buttons to go back to Main Menu and restart the game scene.

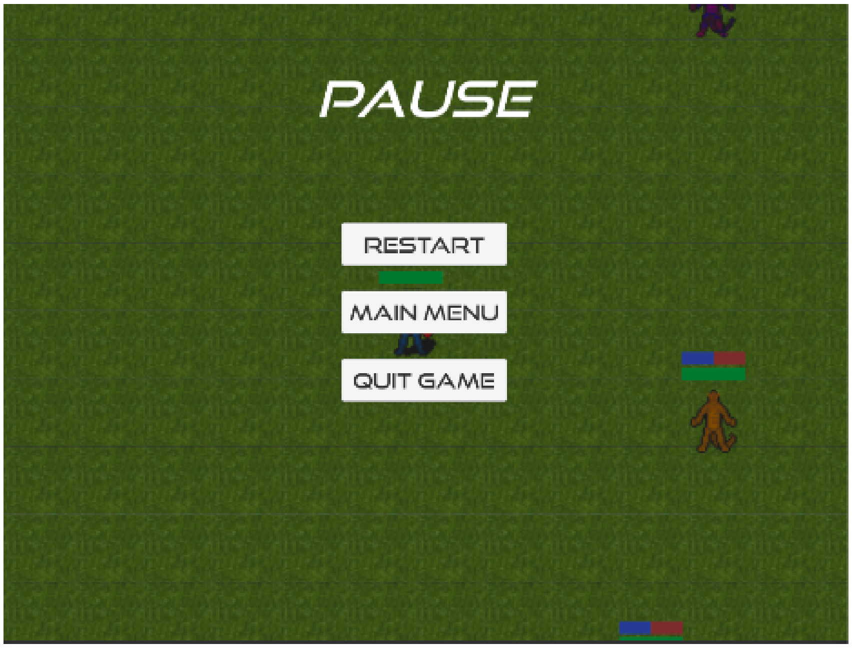

Figure 9: Pause screen screenshot

Figure 10: Victory screen screenshot

5 Emotion, Personality and Culture Classes/Models

This section presents how the chosen models relate with each other in the game context. Therefore, we present the interaction flowchart in Figure 11, in which we can see how personality traits and culture dimensions are combined, resulting in a mental state, translated in a specific behavior. This combination and result will be further explained, mainly on the next subsections. Every use case can be deduced by analysing the table values, and how different combinations can result in different outputs/behaviors, as expected.

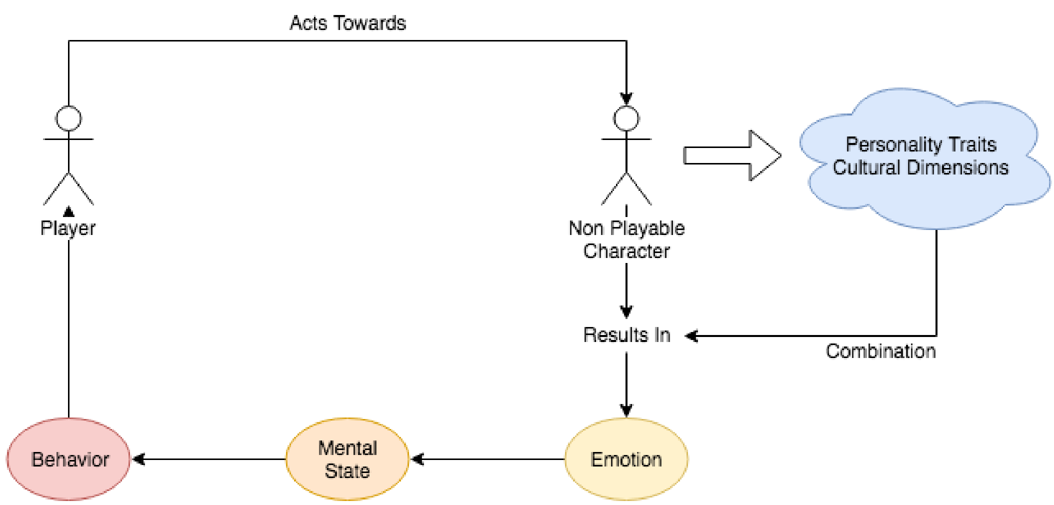

Figure 11: Player-NPC interaction chart

5.1 Defining Emotions

The approach used in our project is an adapted version of the Plutchik's model, in which we only consider the main emotions (normal ones) in a 4-axis structure, as shown in Figure 12. Therefore, the *Emotions* class of our system contains four values: Anger x Fear (A-F); Joy x Sadness (J-S); Anticipation x Surprise (A-S); and Trust x Disgust (T-D). These axis have their values varying from -1 to 1, as shown in tables 7 and 6. The primary, secondary and tertiary dyads values for each emotion, in each axis, are presented in 3, 4 and 5.

Primary Dyads	Combination	Values in Axis
Optimism	Anticipation + Joy	0, 0, 0.5, -0.5
Love	Joy + Trust	0, 0.5, 0.5, 0
Submission	Trust + Fear	0.5, 0.5, 0, 0
Awe	Fear + Surprise	0.5, 0, 0, 0.5
Disapproval	Surprise + Sadness	0, 0, -0.5, 0.5
Remorse	Sadness + Disgust	0, -0.5, -0.5, 0
Contempt	Disgust + Anger	-0.5, -0.5, 0, 0
Aggressiveness	Anger + Anticipation	-0.5, 0, 0, -0.5

Table 3: Primary dyads and its values

Secondary Dyads	Combination	Values in Axis
Hope	Anticipation + Trust	0, 0.5, 0, -0.5
Guilt	Joy + Fear	0.5, 0, 0.5, 0
Curiosity	Trust + Surprise	0, 0.5, 0, 0.5
Despair	Fear + Sadness	0.5, 0, -0.5, 0
Unbelief	Surprise + Disgust	0, -0.5, 0, 0.5
Envy	Sadness + Anger	-0.5, 0, -0.5, 0
Cynicism	Disgust + Anticipation	0, -0.5, 0, -0.5
Pride	Anger + Joy	-0.5, 0, 0.5, 0

Table 4: Secondary dyads and its values

Tertiary Dyads	Combination	Values in Axis
Anxiety	Anticipation + Fear	0.5, 0, 0, -0.5
Delight	Joy + Surprise	0, 0, 0.5, 0.5
Sentimentality	Trust + Sadness	0, 0.5, -0.5, 0
Shame	Fear + Disgust	0.5, -0.5, 0, 0
Outrage	Suprise + Anger	-0.5, 0, 0, 0.5
Pessimism	Sadness + Anticipation	0, 0, -0.5, -0.5
Morbidness	Disgust + Joy	0, -0.5, 0.5, 0
Dominance	Anger + Trust	-0.5, 0.5, 0, 0

Table 5: Tertiary dyads and its values

Mild emotion (0.2)	**Basic emotion (0.5)**	Intense emotion (1.0)
Serenity	**Joy**	Ecstasy
Acceptance	**Trust**	Admiration
Apprehension	**Fear**	Terror
Distraction	**Surprise**	Amazement

Table 6: Main emotions and its values

Intense opposite (-1.0)	**Basic opposite** (-0.5)	Mild opposite (-0.2)
Grief	**Sadness**	Pensiveness
Boredom	**Disgust**	Loathing
Annoyance	**Anger**	Rage
Interest	**Anticipation**	Vigilance

Table 7: Main opposite emotions and its values

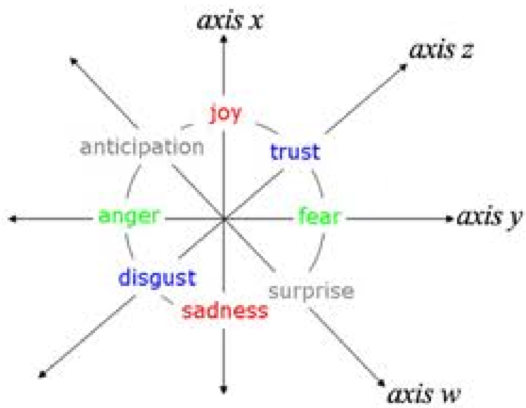

Figure 12: Simplified 4-axis structure - Normal emotions

5.1.1 Defining Cultural Attributes

Böloni et al (2018), on his work "Towards a computational model of social norms", proposed the use of Hofstede's cultural dimensions in social simulations, testing their own social model. Allied with theories about politeness [15] [21], he decided to use four dimensions: time, wealth, dignity and politeness. Even though these were enough for "The Spanish Steps Flower Selling Scam" he presented, they didn't seem enough for our context. Therefore, we decided to use them allied with collectivism and rationality, defined in Hofstede's model.

Even most of these concepts can be mapped using Fuzzy Logic, like what was done in [13], we decided that each NPC would have previously determined culture, i.e. the six dimensions would have its values varying from 0 to 1 (with limits included), as a percentage. For instance, if time dimension was equal to 0.8, this value will multiply the time limit of the current event (between player and NPC), resulting in the maximum time the NPC will spend interacting with the player during this event.

Still, we decided to use an even simpler approach with specific cultural dimensions influencing which emotion the player would have based on the last player's action. If one NPC's dignity value was 0.9, for example, and the player reached his/her personal space, this would result in a high intensity emotion. If the value was 0.2, this would result in an emotion with a minor intensity. Therefore, in the list below, we show how each dimension reflects on the game's content.

1. Time: Influences how fast the NPC will move, multiplying his velocity value;

2. Wealth: Represents how much the player will care about player actions like: money give away, money theft, item give away or item theft. So, if its value is 0.7, this NPC is highly affected by this cited actions.

3. Dignity: As stated before, is related to situations in which the player invades NPC's social, personal or intimate spaces. It also influences how the NPC will be emotionally affected, after he is shot or harmed.

4. Politeness: Represents how much a NPC is affected by a non-polite player approach or conversation;

5. Collectivism: Represents how much a NPC cares about others, when they are inside their public space;

6. Rationality: Its value affects how intense the resultant emotion will be (further explained throughout this report).

5.2 Emotions influence in Behavior

As stated by S. D. Lane, "emotions involve not only thoughts, psychological states, and biological processes but also behavioral propensities. Still others maintain that emotions are socially constructed and learned. Despite disagreement regarding the influence of cognition, psychology, and behavior, most research suggest that emotions are feelings we experience that result from the interaction of psychology, cognitions, and social experience and that they significantly affect how we communicate with others and interpret others' communication."

Thus, we can conclude that emotions have direct influence from communication and behavior. That's why in *Future Falls*, for each different player action towards the NPC, there is a set of possible resultant emotions related to it, as shown in table 8. For instance, if the player shoots and the bullet hits the NPC, the "is_attacking" action is dispatched, resulting in the set composed by "Rage", "Outrage", "Despair" and "Terror".

However, to choose which of these four emotions will be used, we decided to check NPC's cultural dimensions values, relating each action to a specific attribute, as shown in table 9. In this case, we should verify if NPC's dignity value is: bigger than 0.7,

Player's Action	value $\in$ [0, 0.2), [0.2, 0.5), [0.5, 0.7), [0.7, 1.0]
is_attacking	Rage, Outrage, Despair, Terror
is_shooting	Annoyance, Pessimism, Disapproval, Apprehension
is_harming	Anger, Contempt, Unbelief, Fear
is_injured	Anger, Disapproval, Pride, Joy
is_giving_item	Joy, Optimism, Hope, Trust
is_stealing_item	Sadness, Shame, Remorse, Disgust
is_giving_money	Admiration, Love, Sentimentality, Ecstasy
is_stealing_money	Grief, Dominance, Awe, Loathing
is_social	Distraction, Anxiety, Delight, Interest
is_personal	Anticipation, Cynicism, Curiosity, Surprise
is_intimate	Vigilance, Aggressiveness, Submission, Amazement
is_talking_politely	Boredom, Envy, Pride, Serenity
is_not_talking_politely	Pensiveness, Guilt, Morbidness, Acceptance

Table 8: Player's actions and the resultant NPC's emotions, based on the cultural dimension value range

Player's Action	Cultural Dimension
is_attacking	dignity
is_shooting	collectivism
is_harming	collectivism
is_injured	trust_level
is_giving_item	wealth
is_stealing_item	wealth
is_giving_money	wealth
is_stealing_money	wealth
is_social	dignity
is_personal	dignity
is_intimate	dignity
is_talking_politely	politeness
is_not_talking_politely	politeness

Table 9: Player's actions and the related cultural dimension

choosing for the "Terror" emotion; between 0.7 (included) and 0.5, choosing for the "Despair" emotion; between 0.5 (included) and 0.2, choosing for "Outrage" emotion; and less or equal to 0.2, choosing for "Rage" emotion.

5.3 Personality Influence in Emotions

As stated by Revelle and Scherer, "Personality is the coherent patterning of affect, behavior, cognition, and desires (goals) over time and space. Just as a full blown emotion represents an integration of feeling, action, appraisal and wants at a particular time and location so does personality represent integration over time and space of these components [19]. A helpful analogy is to consider that personality is to emotion as climate is to weather. That is, what one expects is personality, what one observes at any particular moment is emotion." [29]

With this statement, allied with a study done by Mohammad and Kiritchenko, we can prove that emotions can help to identify personalities [25], and personality traits can influence emotional behavior. However, as we couldn't find a theory that directly

Fear	Trust	Joy	Surprise	
-1	1	1	-1	O
0	0	0	0	C
0	1	1	1	E
0	1	1	1	A
1	-1	-1	1	N

Table 10: Personality traits influencing positive emotions

Anger	Disgust	Sadness	Anticipation	
0	-1	-1	-1	O
-1	0	1	1	C
0	0	1	-1	E
0	-1	0	-1	A
1	1	1	1	N

Table 11: Personality traits influencing negative emotions

related Plutchik's Wheel of Emotion and OCEAN theory. But these presented studies allowed us to create tables 10 and 11. Therefore, we need to explain how personality and emotion work in the game context.

Each emotion, as explained before, is composed by four values, representing each axis of the simplified Plutchik's Model. During the game's implementation, each emotion was represented by an array, with four positions. The NPC's personality was also represented as an array of five positions, each one representing an OCEAN value, respectively.

Nevertheless, we needed to find a way to combine the personality values with emotion values, deciding if they would influence positively, negatively or not influence at all. Thus, we decided to use the factor tables 10 and 11 proposed by Baffa et al. [14].

Although the previous step resulted in the event emotion (EventEmo), it will be used just to generate a new emotion (NewEmo), based on equation 2. Therefore, each i position of the new emotion will be the sum of the multiplication of three values: event emotion in position i; the personality in position j; and the factor in line j, column i. Then, this value is divided by five, as it was a sum of five values that vary from -1 to 1, resulting in the mean value.

$$NewEmotion_i = \frac{\left(\sum_{j=1}^{5} EventEmotion_i \times p_j \times factor_{ji}\right)}{5}, i \in [1, 4] \tag{2}$$

After that, this new emotion will be added to NPC's current emotion (CurrEmo), but *clamping* the values to fit in range [-1, 1]. This step's calculation is represented in equation 3, in which t represents the current time.

$$(CurrEmo_i)_{t+1} = (CurrEmo_i)_t + Clamp(NewEmo_i, -1.0, 1.0) \tag{3}$$

However, the values from the resultant current emotion may be mixed, not representing any of the listed emotions. To solve this problem, we decided to extract the most influent emotion from the current emotion, following the logic listed below:

1. Iterate through the current emotion values and find the biggest one (comparing absolute values);
2. Iterate again through the emotion values, comparing each one to the biggest value. If they are equal, the influent emotion receives this value, otherwise receives zero.
3. Iterate through the influent emotion values, rounding its values following some rules:
   (a) If there are two values bigger than zero, they both will be considered 0.5;
   (b) Otherwise, it will respect the following rules:
      i. If $Value <= 0.1$, $Value = 0$;
      ii. If $Value <= 0.3$, $Value = 0.2$;
      iii. If $Value <= 0.5$, $Value = 0.5$;
      iv. If $Value > 0.5$, $Value = 1.0$;

5.4 Mental State Influence in Trust Level

As stated before, this most influent emotion is used to decide which is the current NPC's mental state. This attribute is used to decide which behavior the NPC will have, based on the current most influent emotion (table 2). As shown by table 12, each mental state relates to a set of emotions. If the NPC has not been affected by player's action,

Mental State	Related Emotions
Anger	Rage, Anger, Annoyance, Outrage, Envy, Aggressiveness, Dominance
Fear	Apprehension, Fear, Terror, Guilt, Awe, Despair
Trust	Acceptance, Trust, Admiration, Hope, Submission
Disgust	Loathing, Disgust, Boredom, Unbelief, Contempt, Shame
Joy	Serenity, Joy, Ecstasy, Optimism, Love, Sentimentality, Morbidness, Pride
Sadness	Grief, Sadness, Pensiveness, Disapproval, Remorse, Pessimism
Surprise	Distraction, Surprise, Amazement, Delight, Curiosity
Anticipation	Vigilance, Anticipation, Interest, Anxiety, Cynicism

Table 12: Available mental states and its related emotions

Mental State	Trust Level Factor
Anger	-1
Fear	-1
Trust	+1
Disgust	-1
Joy	+1
Sadness	-1
Surprise	+1
Anticipation	-1

Table 13: Mental states and its related factors

his mental stated will be *Neutral*, i.e. the current emotion has its four values equal to zero.

Now that the mental state was defined, we must decrement or increment NPC's trust level based on it. Therefore, we must define what are the mental factors (-1 or 1), that will influence the trust level, as seen in 1. This relation is shown in table 13. If the mental state is *Neutral*, this factor is zero.

6 System Modeling, Requirements, and Implementation

"A System Requirements Specification (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software application. It includes a variety of elements that attempts to define the intended functionality required by the customer to satisfy their different users" [23].

So, this section will explain each of this features and the way we documented and organized our project. Firstly, we will present the *System Modeling*, which is directly related to the way the project was built, also explaining the scripts that were created. Secondly, we will present functional and non-functional requirements, allowing us to understand more what is the system's goal and what are the results it should generate. Finally, we will present other specifications, involving how we tested, documented and controlled the projects versions.

6.1 System Modeling

The proposed system modeling for player and NPCs was defined, during the first part of this project, as seen in Figure 13. However, lot of the planned structure has changed while the game was being developed. Figure 23 (see the end of this chapter) shows the final model of the system, in which we created the classes of *Player* and *Enemy*, but without a global *Character* parent class, as they have very different behaviors and the *Culture* class do not need to be connected with the player (that is controlled by a human).

Besides these classes, there is a *Region* class, that allied with its *dominantCulture* attribute, can generate a *Tradition* instance, which define the NPC's habits after they spawned. For instance, characters that live in a region that has a hot weather wear few clothes. They may also drink lots of liquid and eat less spicy and hot food. The *Ecosystem* class was defined to generate the fauna and flora of the region which will be spawned. This classes models are shown in Figure 14.

Even though these last three classes were defined, they were not used in our system, as we focused on the player and NPC reaction. They would be used when generating the random maps, using PCG strategies.

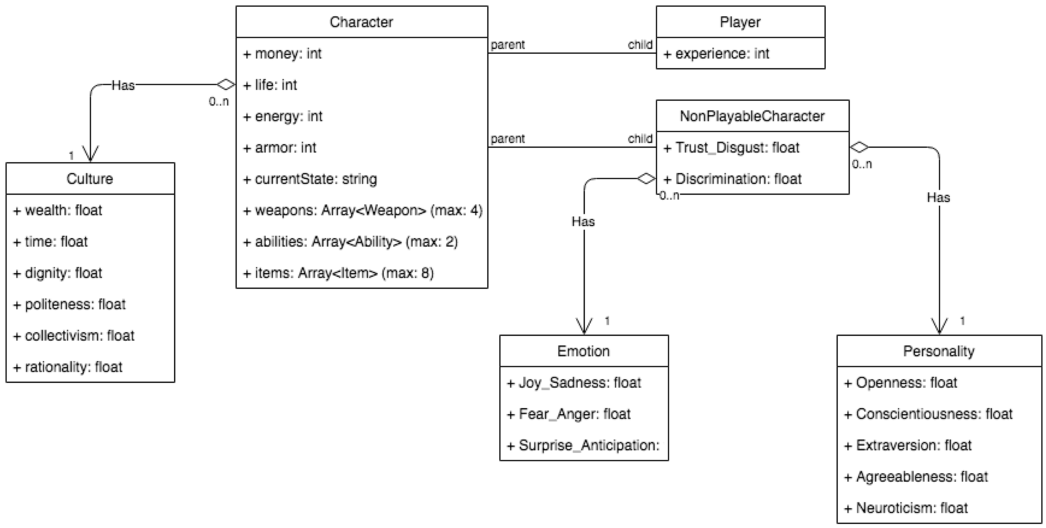

Figure 13: *Future Falls'* characters class diagram (proposed)

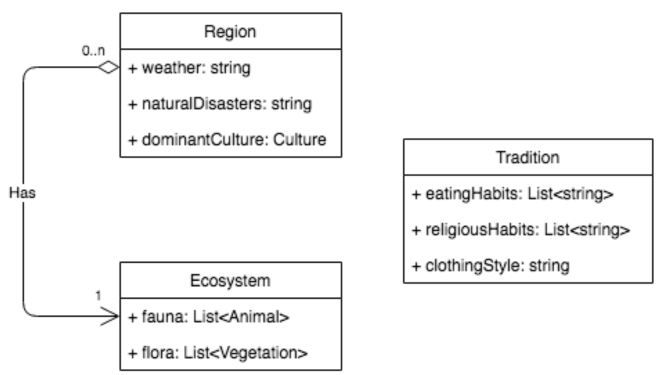

Figure 14: Region and Tradition Classes

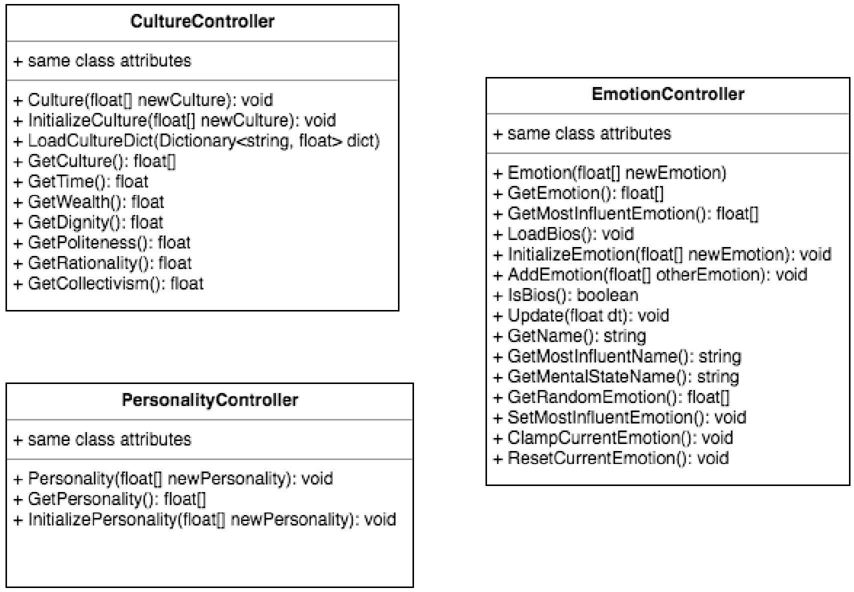

Figure 15: Non behavior controllers

To control the behaviors for each class, we needed *Controllers*, which represent the scripts created for the project. There are two types of Controller: the ones that extend from *MonoBehavior*, a Unity class that brings lots of built-in functions, like *Start*, *Update*, *OnCollisionEnter2D*, and others (shown in Figure 24, at the end of this chapter); and the ones that doesn't extend any other class (shown in Figure 15). Therefore, the three main controllers are: PlayerController, EnemyController (NPC controller) and GameController.

The first will be responsible for receiving player's inputs and outputs, transforming it in gameplay actions. For example, inside the *Update* function, *UpdateMovement* is called, verifying when the player presses the movement keys, incrementing or decrement player's positions, following the equation 4. The example code is shown in Figure 16.

$$position_{t+1} = position_t + movementSpeed \times dt \tag{4}$$

The second will be responsible for update enemies' behavior, its *trustLevel*, its healthBar, and other properties (code snippet shown in Figure 17,). The player proxemics will be updated by *ProxemicsController*, instead of the *EnemyController*. In Figure 19, we can see how proxemics are implemented (green and red circumferences).

The third will be responsable for spawning NPCs thourghout the map, calculate the

```
/// <summary>
/// Updates the movement.
/// </summary>
/// <param name="dt">Delta time.</param>
void UpdateMovement(float dt) {
    float moveHorizontal = Input.GetAxis("Horizontal");
    float moveVertical = Input.GetAxis("Vertical");
    Vector2 position = playerBody.position;

    position.x += moveHorizontal * movementSpeed.x * dt;
    position.y += moveVertical * movementSpeed.y * dt;

    playerBody.position = position;
}
```

Figure 16: *UpdateMovement* function, from *PlayerController*

```
/// <summary>
/// Updates the normal movement.
/// </summary>
/// <param name="dt">Delta time.</param>
void UpdateNormalMovement(float dt) {
    if (normalStateTimer > maxWalkingTime) {
        currentState = "stopped";
        normalStateTimer = 0;
    }

    Vector2 position = enemyBody.position;

    position.x += direction.x * movementSpeed.x * dt;
    position.y += direction.y * movementSpeed.y * dt;

    enemyBody.position = position;

    if (!engagedInAction) {
        normalStateTimer += dt;
    }
}
```

Figure 17: *UpdateNormalMovement* function, from *EnemyController*

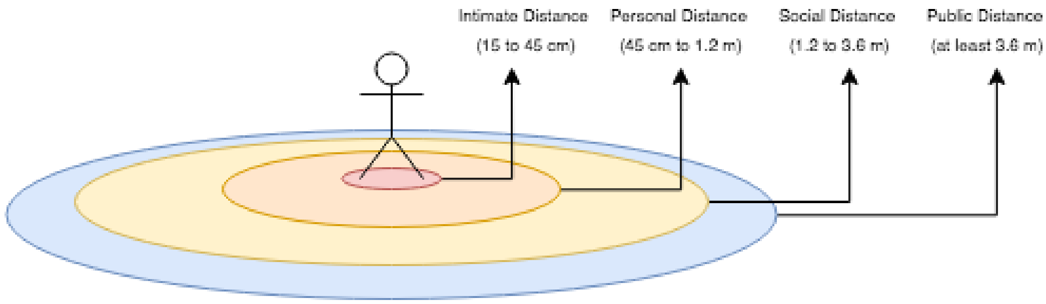

Figure 18: Proxemics - Interpersonal distances proposed by Hall

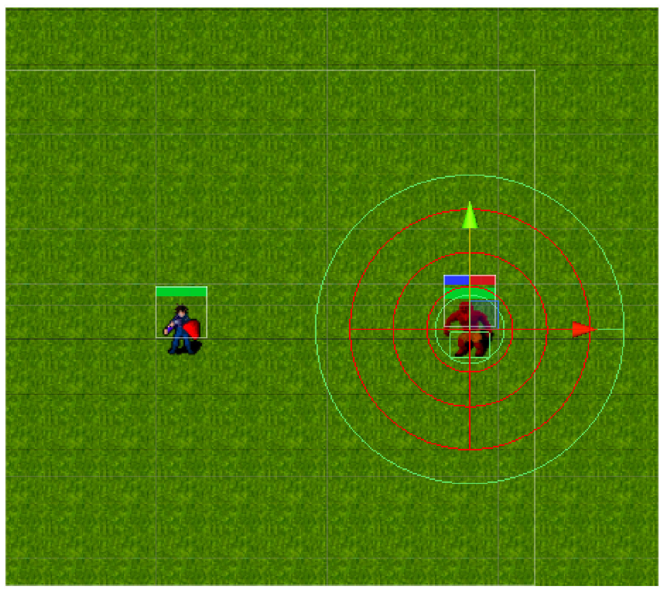

Figure 19: Proxemics in game

mean trust level, and other general functionalities. We can see *SpawnEnemies* function (with enemies meaning NPCs) in Figure 20.

6.2 Functional requirements

These are requirements that define what the system should do. In other words, what are the system inputs, outputs, what the system should store, the computations it should make and all the timing and synchronization related to this elements [34]. Therefore, knowing that the system we should build is a *Role Playing Game*, its functional requirements are:

1. System inputs and outputs: The player must have full control towards players

```
/// <summary>
/// Spawns enemies in random positions.
/// Instantiates "Enemy" prefab
/// </summary>
void SpawnEnemies() {
    for (int i=0; i < totalEnemies; i++) {
        GameObject newEnemy = Instantiate(enemyPrefab, null);
        Vector2 pos = newEnemy.transform.position;
        pos.x = pos.x + i*0.5f;
        pos.y = pos.y + i * 0.5f;
        newEnemy.transform.position = pos;

        spawnedEnemies.Add(newEnemy);
    }
}
```

Figure 20: *SpawnEnemies* function, from *GameController*

movement and actions. Also, they must be able to move in the game space freely, just being stopped when arriving in the edge of the map or when colliding to a NPC. He/she must behave differently, depending on the possible player actions, as presented in previous sections;

2. Stored values: The player will be able to save the game anytime, except in a Dungeon or during a battle. Each time the user hits the save button on the game inventory interface, some important values are saved in a ".txt" file, named as "GameLog_*TIMESTAMP*", with *TIMESTAMP* being the time in which the game was saved. The values registered on this file are:

   (a) The life he/she had during the save;

   (b) The energy he/she had during the save;

   (c) The total money he/she had during the save;

   (d) The position he/she was in a specific region during the save;

   (e) The weapons the player is currently using;

   (f) The weapons and items stored in his/her inventory.

3. Computations: A game is a real-time application, which means it is constantly being updated throughout the time the player interacts with it. Therefore, all the keys (keyboard or joystick) the player pushes are identified, which can result in an specific action. This action is processed as a state machine, identifying which behavior the NPC will have as a response to that behavior. The logic behind that

computation was already explained, involving emotion, personality, proxemics and culture factors.

6.3 Non-functional requirements

These requirements are defined by the constraints that must be imposed on the design of the system [34]. For instance, in this subsection we will talk about: frame rate, response time, immersion and fun.

1. Frame rate: The game application should run in at least 85 frames per second, and the 3D application should run in 105 frames per second, when running in the minimal requirements. As we couldn't test in various specifications, the defined ones are:
   (a) RAM: 8GB;
   (b) Hard Disk: 869 MB;
   (c) Processor: Intel core i7;
   (d) Video Card: Intel HD Graphics 6000, 1.5 GB RAM;
2. Response time: The game must take 5 to 10 seconds to start properly, and the Non Playable Characters should respond to player's actions immediately;
3. Immersion and fun: As we couldn't test this game with the public, this factors couldn't be tested;
4. Cost: All game images, fonts and animations were downloaded from free repositories, but the models for the 3D application were bought from *Unity Asset Store* [32], for the price of almost 50 dollars.

6.4 Implementation, Documentation, Tests, and Platforms

During this project's implementation, some features and details could not be done. However, most of the planned feature were implemented and tested, resulting in a game with death and victory conditions, along with NPC interaction. In this section, we summarize the main aspects concerning the following questions: tests and platforms used to distribute the game executable; how we implemented and documented the code, along with the game engine used; and how we controlled our project's versions. We present these points as follows:

1. Tests: We tested the system always trying to change its inputs. For instance, to test the various possible NPC's reactions, we would first, instead of basing ourselves in random emotions, cultures and personalities, try some fixed values, and verify which mental state was generated by player's action. In the other hand, to verify player's inputs' reliability, we just entered with the keyboard and joystick inputs, checking if the visual feedback corresponded with planned;

2. Implementation and documentation: The code was done in C# language. Figure 21 illustrates the code comment style we used in this work. The scripts were organized in relation with the classes defined in the system modeling;

3. Platforms for testing: The game has its executable file available for Windows and Mac operational systems;

4. Platforms for development: *Unity* is the engine we used to develop most of the game, as told in previous sections.

5. Platform for access of the resources: We used *Github* to control the project versions (scripts, 3D models and images), and *Google Drive*, and our personal computers, to store the related works (papers, books and reports) found.

```
/// <summary>
/// Updates the emotion by event.
/// </summary>
/// <param name="eventEmotion">Event emotion.</param>
void UpdateEmotionByEvent(float[] eventEmotion) {
    float[] newEmotion = new float[4];
    float[] p = personality.GetPersonality();
    float[,] pFactors = Personality.PositiveFactors;
    float[,] nFactors = Personality.NegativeFactors;

    // Generate new emotion based on Personality Traits and Factors
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 5; j++) {
            if (eventEmotion[i] > 0)
                newEmotion[i] += eventEmotion[i] * p[j] * pFactors[j, i];
            else
                newEmotion[i] += eventEmotion[i] * p[j] * nFactors[j, i];
        }
        newEmotion[i] = newEmotion[i] / 5;
    }

    // Add new generated emotion
    emotion.AddEmotion(newEmotion);
    // Clamp after emotion add
    emotion.ClampCurrentEmotion();
    // Set the most influent emotion
    emotion.SetMostInfluentEmotion();
}
```

Figure 21: Example of code commentary

7 Conclusions

Since the beginnning of this project, our main goal was to create a 2D game that could simulate cutural behavior in some level. Therefore, we used well-known models and theories, that allied with some equations and methods for artificial intelligence, resulted in a complex logic, which would just relate to NPCs behavior.

By far the most difficult part of this project was this behavioral simulation. We always had to think in the best ways to combine emotional factor, personality traits, interpersonal space theory and cultural dimensions. This challenge allowed us to evolve as researchers and made us rethink of some goals, restructuring the project as a whole.

In the end, we developed a action/adventure game, which focuses on the player's actions and the NPC's reactions. The main goal remains the same: maximize the mean of NPC's trust level. However, some actions needed to be simplified, like the item/money exchange, which was reduced to a button in the action menu, or even the player and NPC conversation, also reduced to a "point and click" action (seen on Figure 22).

We decided to focus on the NPC behavior itself, as the goal was to test the model, and see if these theories together would result in something academically relevant. Fi-

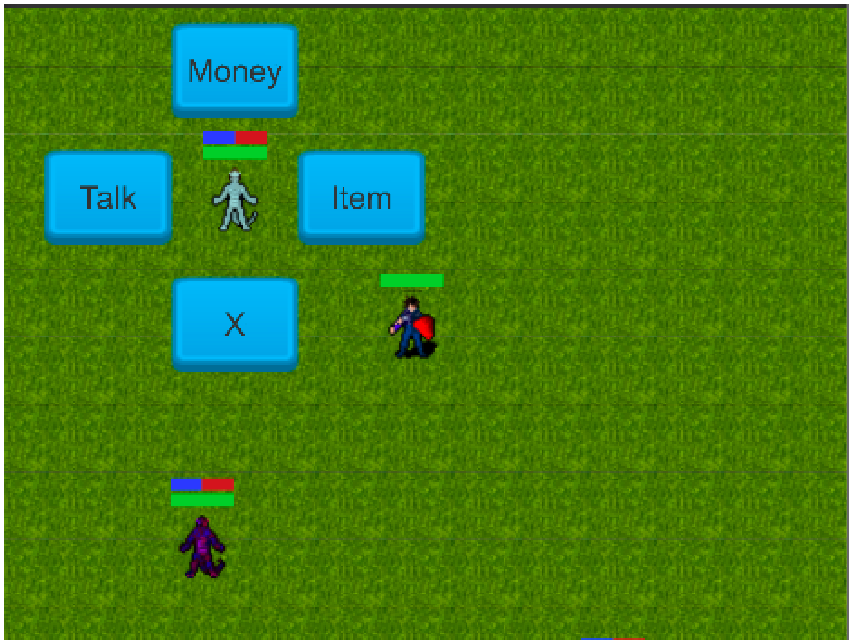

Figure 22: Interactive game menu

nally, we decided to divide our application in two Unity scenes: the first is a game, in which the player would interact and try to win the game (reach 80% or above in the mean trust level), or just have fun discovering the possible interactions; the other is a scene where a 3D model would react to specified key commands, simulating a player action, resulting in a behavior more expressive that the ones in the first scene.

This decision helped us have our first experience generating 3D emotional reactions, while testing well structured scripts in a 2D simple game. Clearly, we expected a great experience using Unity 3D Engine, even though we had implemented other projects with it, always discovering new ways to implement some feature, or the best approaches when programming a game application.

8 Maintenance

Even though our project presents an unique approach to culture simulation in games, there are still lots of other ways to specify cultural behaviors of game characters. For instance, we could base ourselves on other cultural dimension, different from Hofstede's, basing ourselves on the definition of culture, by Schimitz [12]. Also we could study more deeply how culture characteristics can be mapped into a game. Furthermore, we need to improve the simple way the characteristics are represented (i.e. something better than simple values from 0 to 1). A possible approach to this later problem is to use one function for each different characteristic, like what was done in [16], applied to a video game context.

Moreover, other emotion and personality theories could be used, as other projects like *FAtiMA* did, using the OCC theory [17] to influence virtual agents behavior, making it generic to use even appraisal theories [18]. This approach was used in a game called *FearNot!* [35]. Other questions can also be investigated, such as the exploration of stronger relations in the way personality and emotions affect people's cultural dogmas. In this later context, we can explore how a virtual agent could change his personality depending on an important event, or change even his/her social beliefs, affecting his/her future actions.

During the design of this project, we thought a lot about how we could make the *prejudice level* influence the interactions between the player and the NPCs. This characteristic should not be related to an individual, but to the current global situation of the region in which he/she lives. For instance, a character may discriminate a culture that he/she never interacted before, but he/she can learn with this first experience and change his/her own discrimination value.

In the end, our project brings lots of solutions related to NPCs behaviors based on cultural characteristics, but also raises lots of questions, greatly contributing to research in game narrative and worldbuilding. Also, the model created can be used in different virtual agents, not being limited to video games.

References

[1] Newzoo: Game industry growing faster than expected, up 10.7% to $116 billion 2017. https://venturebeat.com/2017/11/28/newzoo-game-industry-growing-faster-than-expected-up-10-7-to\-116-billion-2017/, 2017. Acessed: 18/11/2018.

[2] Binary domain max trust guide – consequence system. https://segmentnext.com/2012/02/26/binary-domain-max-trust-guide-consequence-system/, 2012. Acessed: 19/11/2018.

[3] Rpgs took over every video game genre. https://www.ign.com/articles/2012/12/12/rpgs-took-over-every-video-game-genre, 2012. Acessed: 21/11/2018.

[4] This engine is dominating the gaming industry right now. https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/, 2016. Acessed: 20/11/2018.

[5] Video game industry is booming with continued revenue. https://www.cnbc.com/2018/07/18/video-game-industry-is-booming-with-continued-revenue.html, 2018. Acessed: 15/11/2018.

[6] How procedural generation took over the gaming industry. https://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/, 2014. Acessed: 20/11/2018.

[7] The brazilian gamer — 2017. https://newzoo.com/insights/infographics/the-brazilian-gamer-2017/, 2017. Acessed: 15/11/2018.

[8] Mid-sized developers argue that the market might be saturated with games. https://wccftech.com/aa-developers-market-saturated-games/, 2017. Acessed: 19/11/2018.

[9] 7 exciting trends from the future of video games. https://www.oxford-royale.co.uk/articles/7-exciting-trends-future-video-games.html, 2017. Acessed: 21/11/2018.

[10] The roots of open-world games. https://www.gamesradar.com/the-roots-of-open-world-games/, 2018. Acessed: 18/11/2018.

[11] The history of elite: Space, the endless frontier. http://www.gamasutra.com/view/feature/3983/the-history-of-elite-space-the-.php, 2009. Acessed: 18/11/2018.

[12] Schimitz A. *Cultural Intelligence for Leaders*. 2012.

[13] Atifa Athar, M.Saleem Khan, Khalil Ahmed, Aiesha Ahmad, and Nida Anwar. A fuzzy inference system for synergy estimation of simultaneous emotion dynamics in agents. *International Journal Scientific & Engineering Research*, 06 2011.

[14] Feijó B. Baffa A., Sampaio P. Dealing with the emotions of non player characters. *SBGames 2017*, 2017.

[15] P. Brown, S.C. Levinson, Cambridge University Press, S.C. Levinson, J.L. Gumperz, J.J. Gumperz, P. Drew, M.H. Goodwin, and D. Schiffrin. *Politeness: Some Universals in Language Usage*. Studies in Interactional Sociolinguistics. Cambridge University Press, 1987.

[16] Khan S.A. Streater J. Fiore S.M. Bölöni L., Singh Bhatia T. Towards a computational model of social norms. *PLoS ONE Vol 13, No. 4, 2018. [DOI: 10.1371/journal.pone.0195331]*, 2018.

[17] A. Herzig C. Adam and D. Longin. A logical formalization of the occ theory of emotions. 2009.

[18] Paiva A. Dias J., Mascarenhas S. Fatima modular: Towards an agent architecture witha generic appraisal framework. 2012.

[19] J.-M Fellous and Michael Arbib. *Who needs emotions?: The brain meets the robot.* 01 2012.

[20] H. Games. No man's sky (digital game). 2016.

[21] Yueguo Gu. *Modern Chinese Politeness Revisited*, pages 128–148. Palgrave Macmillan UK, London, 2011.

[22] G. Hofstede and M. Minkov. *Cultures and organizations: software for the mind.* McGraw-Hill Professional, 2010.

[23] Inflectra. What is a system requirements specification (srs)? `https://www.inflectra.com/ideas/topic/requirements-definition.aspx`, 2019. Acessed: 23/07/2019.

[24] P. John and S. Srivastava. *The big-five trait taxonomy: History, measurement, and theoretical perspectives.* In L. A. Pervin & O. P. John (Eds.), 1999.

[25] Saif M. Mohammad and Svetlana Kiritchenko. Using nuances of emotion to identify personality. *CoRR*, abs/1309.6352, 2013.

[26] Mojang. Minecraft (digital game). 2009.

[27] Nintendo. The legend of zelda: A link to the past (digital game). 1991.

[28] Plutchik R. *The nature of emotions.* American Scientist, 2001.

[29] William R Revelle and Klaus Scherer. *Personality and emotion*, pages 304–305. Oxford University Press, 2009.

[30] Scott Rogers. *Level Up! The Guide to Great Video Game Design.* Wiley Publishing, 2nd edition, 2014.

[31] Sega. Binary domain (digital game). 2012.

[32] Unity Store. Unity asset store - polygon scifi package. `https://assetstore.unity.com/packages/3d/environments/sci-fi/polygon-sci-fi-city-pack-115950`, 2019. Acessed: 19/05/2019.

[33] PlayABL Studios. Façade. 2005.

[34] R. Laganière T. C. Lethbridge. *Object-Oriented Software Engineering: Practical software development using UML and Java.* McGraw-Hill Education, 205.

[35] Enz S. Vannini N. and Sapouna M. “fearnot!”: a computer-based anti-bullying-programme designed to foster peer intervention. 2010.

Appendices

A Models

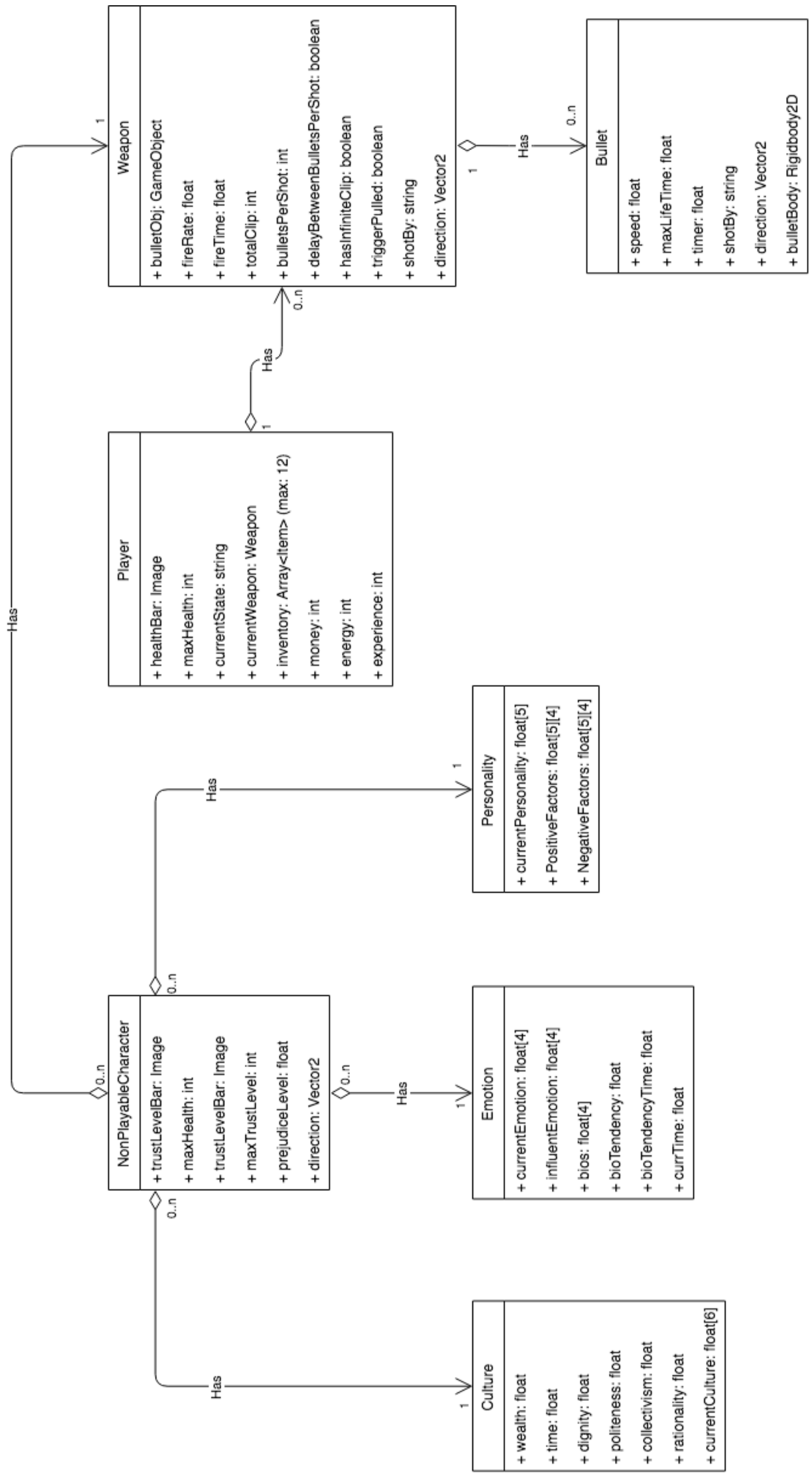

Figure 23: *Future Falls'* characters class diagram (used)

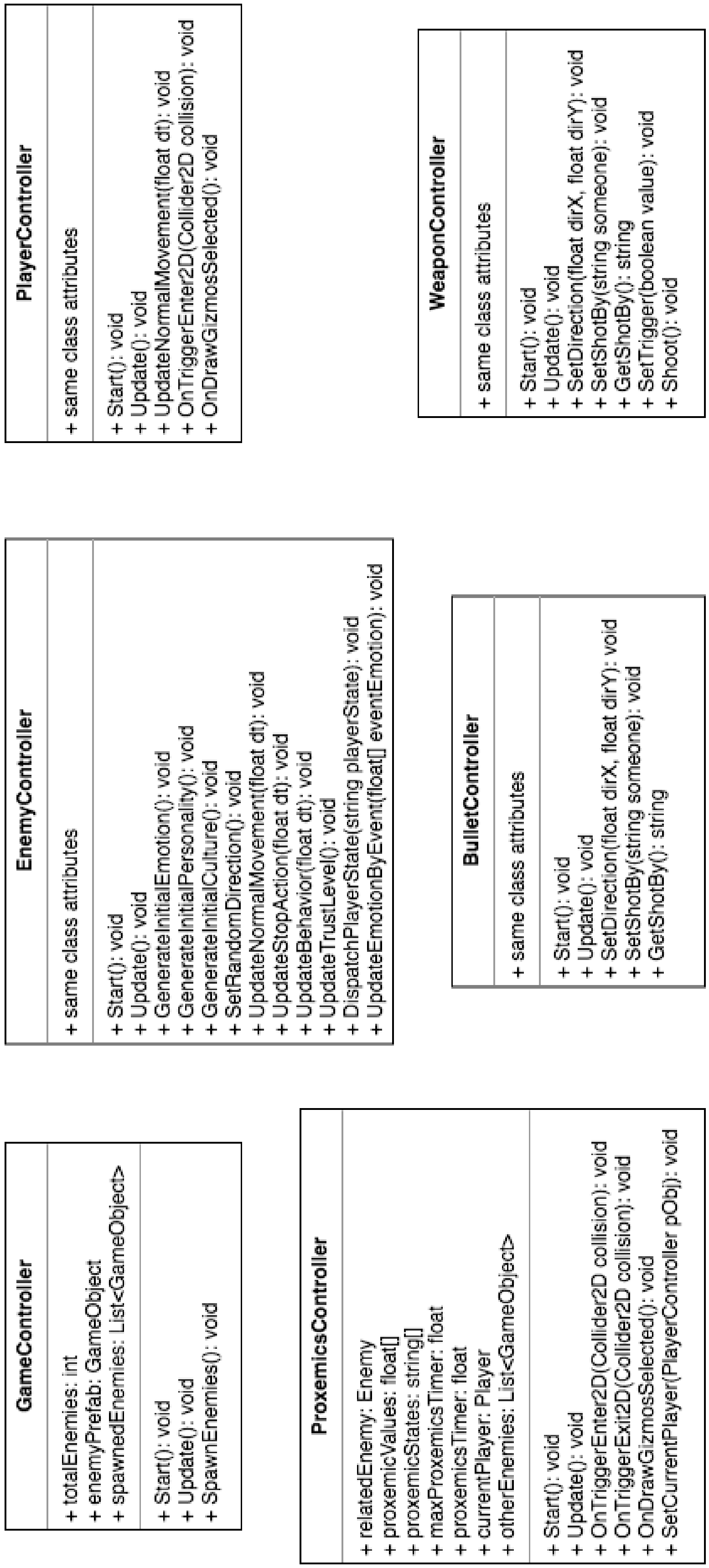

Figure 24: Behavior controllers