CoBelievable – The Effect of Cooperation in Believability

Ana Patrícia Fred Filipe

Thesis to obtain the Master of Science Degree in Information Systems and Computer Engineering

Supervisors: Prof. Carlos António Roque Martinho
Professor João Miguel de Sousa de Assis Dias

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Tiago João Vieira Guerreiro

December 2015
Abstract

As Games Technology evolves, bringing us ever more realistic graphics and animations, there is the need to evolve the game characters’ Artificial Intelligence (AI), creating more interesting, immersive, believable characters, that the player can relate to.

In this work, we address this issue focusing our efforts in exploring the manifestation of personality as a way to create interesting, believable AI characters. Towards this goal, we created an agent model that manifests different personalities based on a conflict-handling model. We designed an agent model based on a Behavior Tree architecture, with the extension of utility-based evaluation and dynamic re-ordering of the behaviors of the tree. The designed model, as well as a custom scenario test-bed, were implemented in Unreal Development Kit (UDK).

We conducted experimental analysis of the model with user testing to verify if the personalities could be effectively conveyed. Experimental results showed the adequacy of the proposed solution, proving that the model could yield distinguishable and believable personalities.

Keywords: Virtual game agents, conflict model, behavior trees, companion AI
Resumo

À medida que a tecnologia da indústria de jogos evolui, oferecendo-nos gráficos e animações cada vez mais realistas, torna-se aparente a necessidade de evoluir de igual modo a Inteligência Artificial (IA) das personagens dos jogos, de forma a criar personagens mais interessantes, imersíveis e credíveis, com as quais o jogador possa empatizar.

Neste trabalho, abordamos este tema focando-nos na exploração da manifestação de personalidade como forma de criar personagens de IA credíveis e interessantes. Neste sentido, desenvolvemos um modelo de agente que manifesta diferentes personalidades com base num modelo de resolução de conflitos.

O modelo de agente baseia-se numa arquitectura de Árvores de Comportamento, que complementamos com um mecanismo de avaliação de Utilidade, bem como uma reordenação dinâmica dos comportamentos que compõem a árvore.

Implementámos o modelo desenhado, juntamente com um jogo criado de raiz para servir de cenário de teste, no motor de jogos Unreal Development Kit (UDK).

Realizámos testes experimentais com utilizadores de forma a verificar se as diferentes personalidades eram distinguidas e correctamente compreendidas.

Os resultados experimentais demonstraram a adequação da solução proposta, provando que o modelo permite produzir personalidades distintas e credíveis.

**Palavras-chave:** Agentes virtuais, modelo de conflicto, árvores de comportamento, IA em jogos
Contents

Abstract ................................................................. iii
Resumo ................................................................. iv
Contents ................................................................. vi
List of Figures ......................................................... viii
List of Tables ......................................................... ix

1 Introduction ......................................................... 1
  1.1 Motivation and Problem Description .............................. 1
    1.1.1 Cooperation, Conflict and Personality ......................... 2
    1.1.2 Goals and Work Outline ...................................... 3
  1.2 Original Contributions .......................................... 3
  1.3 Thesis Outline .................................................. 4

2 Background and State of the Art ................................ 6
  2.1 Conflict Theory .................................................. 6
    2.1.1 Conflict Definition ........................................ 7
    2.1.2 Conflict Models ........................................... 7
    2.1.3 Conflict-handling Modes .................................. 7
    2.1.4 Conflict in Games .......................................... 9
  2.2 Believability ..................................................... 9
    2.2.1 Personality ................................................ 11
    2.2.2 Emotion .................................................... 12
  2.3 Cooperative Agents ............................................. 13
    2.3.1 Cooperative Agents in Research ........................... 13
    2.3.2 Cooperative Agents in the Games Industry ................ 14
  2.4 AI Models in the Game Industry ................................ 15
  2.5 Concluding Remarks ........................................... 17

3 Approach and Architecture ...................................... 19
  3.1 Experience Design ............................................. 19
  3.2 Agent Architecture ........................................... 21
    3.2.1 Behavior Tree ........................................... 22
# List of Figures

2.1 Conflict handling modes of the dual-concern model [4]. ........................................ 8  
3.1 Cooperative bot leaving all the goods for the player. .............................................. 20  
3.2 Agent Model Process. ......................................................................................... 21  
4.1 Bot initialization Kismet sequence. ........................................................................ 28  
4.2 Beginning of the scenario. ................................................................................... 29  
4.3 Basic task object structure. ................................................................................. 30  
4.4 Behavior Tree Kismet sequence. .......................................................................... 31  
4.5 Pick-up Items. ..................................................................................................... 32  
4.6 Platform puzzle revealing bonus item. .................................................................. 33  
4.7 Platform puzzle revealing hidden section of the room. ....................................... 33  
4.8 Initial corridor with platform-activated door mechanism and enemies. ............. 34  
4.9 Markers for Hiding Spots and Exploration Node, and Ammo Kit prefabs in room B. 35  
4.10 Final room. ...................................................................................................... 35  
4.11 First room after corridor. ................................................................................... 36  
4.12 Layout of the scenario level. Green squares represent puzzle platforms. Yellow circles represent the keys the player must gather. Obstacles are presented as blue elements. . 37  
5.1 Demographic information on the testers age. ...................................................... 40  
5.2 Demographic information on the testers gender. ................................................. 41  
5.3 Demographic information on the testers games experience. ................................ 41  
5.4 Demographic information on the testers experience with FPS games. ................ 41  
5.5 Demographic information on the testers gaming habits. .................................... 42  
5.6 Scatter plot with the results for the user mapping of each of the Bot personalities. 43  
5.7 Heat map of the results for the users mapping of the perceived Bot personalities. . 44  
5.8 Confusion matrix for the quadrant matching. ....................................................... 45  
5.9 Confusion matrix just for the Cooperativeness discerning. .................................... 45  
5.10 Confusion matrix just for the Assertiveness discerning. ....................................... 45  
5.11 Classifier regions for the Gaussian distribution approach. .................................. 48  
5.12 Confusion matrix for the Gaussian classifier approach. .................................... 48  
5.13 Classifier regions for the Parzen Windows distribution approach. .................... 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.14 Confusion matrix for the Parzen Windows classifier approach</td>
<td>49</td>
</tr>
<tr>
<td>5.15 Histogram of players’ favourite bot personalities</td>
<td>50</td>
</tr>
<tr>
<td>A.1 First part of the web survey</td>
<td>60</td>
</tr>
<tr>
<td>A.2 Printed part of the survey</td>
<td>61</td>
</tr>
<tr>
<td>A.3 Second part of the web survey</td>
<td>62</td>
</tr>
</tbody>
</table>
List of Tables

5.1 p-value results for the Kolmogorov–Smirnov test for the Cooperativeness dimension. . . . 46
5.2 p-value results for the Kolmogorov–Smirnov test for the Assertiveness dimension. . . . . 46
5.3 p-values Results for the t-test for the Cooperativeness dimension (rows represent the assigned personality, columns represent the perceived personality). . . . . . . . . . . . . . . 47
5.4 p-values Results for the t-test for the Assertiveness dimension (rows represent the assigned personality, columns represent the perceived personality). . . . . . . . . . . . . . . 47
C.1 Verbal evaluation of the bot personalities. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
Chapter 1

Introduction

1.1 Motivation and Problem Description

In recent games there is still the need for new ways of designing interesting companion characters, as the current ones usually look too artificial and lacking in personality in the players’ eyes, simply following pre-scripted behaviors. Even in the highly-acclaimed and beautifully-illustrated game Skyrim, the companion characters sometimes exhibit awkward behavior choices, and all the different characters act the same, devoid of distinct personality when accompanying the player. It is only through the different voice-acted emotes that the different personalities can be conveyed to the player.

This problem is not related to the character’s correct and efficient performance (or lack of it), but to the concept of believability.

Believability relates to the illusion of life, looking similar to a being with ability of thought, feelings and interacting with others. When the player perceives that the companion character has its own interests and personality, the agent is then considered “believable” [1]. This way, the overall game experience is much improved, providing greater immersion with the feeling of having real company, presenting different and less predicable interactions [2].

Personality, being a very important aspect of designing believable game agents, has a great impact in social interactions, where it can be noticed most. Moreover, in order to design an interesting companion agent, it is important to consider the expression of its personality, not only through interactions with other agents, but also with the player, who plays the central role in a game. This makes it essential for agents to exhibit appropriate teamwork interactions, and, particularly, cooperate with the player in a meaningful way, so that the personality traits can be clearly conveyed[3]. Cooperation means that the involved parties work together for their common purpose. This setting provides a scenario for significant human-bot interactions.

Taking this into consideration, the problem addressed by this thesis relates to the expression of personality through cooperative behaviors, as a means of increasing the believability of the character in its interactions with the player.
1.1.1 Cooperation, Conflict and Personality

Although in cooperation a group of agents tries to act together for a common benefit, when personal agendas interfere with each other or their common goal, a conflict situation may arise. In every society, an important aspect of cooperation is the behavior choice when faced with a conflict of interests. According to Kenneth Thomas [4], there is a general definition of conflict relating to the interdependence between the parties, perception of incompatibility of their concerns and how they interact, and is defined, in his words, as “the process which begins when one party perceives that another has frustrated, or is about to frustrate, some concern of his”. This kind of situation arises very frequently in games where some kind of cooperation with a character is required, since often there is a divergence of goals between the player and the AI character. Most of the times, when conflict arises in those games, the player is left with a feeling of frustration, since AI team-mates will most often disregard the player and not give him any feedback, pursuing their own scripted goals in a cold robotic fashion. However, in a real environment, the participating parties manifest their personalities in different ways considering the level of interest on the other party’s goals and welfare.

We believe that a very important aspect to designing believable characters is effectively expressing their personality, and we see the player-AI interactions in conflict situations as a great opportunity to show the characters personalities. This way, we pursue the direction of developing bots personalities as a means to improve their believability in this social scenario of exhibiting different ways of handling conflict.

Researchers have represented these interaction styles as either pro-self social orientations, i.e. motivated by their own individual and competitive goals, or cooperative and altruistic pro-social orientations, to maximize both their own outcomes as well as the other party’s.

Another view for conflict management is the dual-concern model regarding the varying degrees of concern that one has for their own outcomes as well as for the other party’s outcomes [5]. This latter model was proposed by Kenneth Thomas in 1976, and sets 5 main conflict management styles: avoiding, accommodation, competition, compromise and collaboration [4]. This model, although simple, presents a solid foundation to build on and try to map interaction behaviors that are consistent with the conflict management style, and then yield a way to dynamically create an agent with personality based on the position on the two axis of the model: Assertiveness – concern for self, Cooperativeness – concern for the other. In order to address the stated research problem, we focus on the expression of personality in conflict resolution situations, mainly arising from goal incompatibility in a cooperative context.

As we mentioned before, this social environment where the bot and player’s different ideas and necessities clash seems a great opportunity to present the expression of personality to the player, and we believe that through expression of personality the believability of a character is greatly improved. This way, we are convinced that our approach to the problem of creating believable characters can be successful.
1.1.2 Goals and Work Outline

Our main goal in this work is to create a virtual agent that expresses personality through its behaviors in a cooperative context. In particular, we explore a conflict model in order to regulate the different cooperative behaviors of our agent, by correlating conflict resolution modes to different personalities. By doing this, we attempt to provide the gaming industry with a different approach to cooperation believability.

In order to solve our problem of creating a believable AI companion expressing personality through cooperative interactions, we look at how conflict strategies influence behavior, exploring a conflict model approach.

We designed and implemented a framework for an agent with a varying cooperation personality, yielded by its behavior parameterization depending on the choice position on the two axis of concern for self and concern for the other represented in Thomas’ conflict management model. We attempted to design interesting virtual characters with believable cooperation interaction by exploring the different ways the agents can interact with the player and exhibit different behaviors.

In the interest of having a suitable environment for testing our model, a game was created, in a First Person Shooter (FPS) style, with simple puzzle elements, which require cooperation between player and AI companion.

Finally, user tests were conducted and corresponding results were analyzed, in order to verify the quality of the model and validate the proposed solution.

1.2 Original Contributions

In the context of this work, the following contributions were achieved:

- **Conflict-based Personality Model using Behavior Trees.** With the goal of testing personality perception in a cooperative environment, we created an agent manifesting different personalities parameterized by Thomas’ Dual-Concern Conflict model [4].

- **Behavior Trees framework for UDK Controllers Configurable by Kismet.** UDK by default does not support agent controllers that use Behavior Trees, and the community-based contributions are quite limited in this respect. By creating a controller extension that communicates with separate Behavior Tree system to make decisions, we provide a new tool for any developer looking into integrating the Behavior Tree model into their AI’s. This new Behavior Trees system includes a Kismet-based configuration tool, so that adjusting the Behavior Tree’s components is very easy, in a way that even game designers with no programming experience can configure the AIs behaviors. This framework can be easily extended with new behaviors, by extending the existing ones.

- **UDK Testbed for Agents with Cooperative Elements.** Following the previous contribution, we create a testbed in UDK for testing our cooperative agents. This testbed required new elements such as cooperative challenges, exploration markers, and objective markers.
1.3 Thesis Outline

The rest of this document is structured as follows.

In Chapter 2 we define the basic concepts needed for the understanding of our work and describe the state of the art on personality and conflict models. In Chapter 3 we present the methodology developed to address our problem. Chapter 4 describes the implementation of the solution proposed in Chapter 3. In Chapter 5 we describe the experimental setup and present and discuss the experimental results. In Chapter 6 we present concluding remarks and pointers for future work.
Chapter 2

Background and State of the Art

In order to better contextualize our work, in this section we will delve into the topics that this work focuses on, providing an overview of related work.

As stated before, our goal is to design a believable agent that expresses personality through its behavior in cooperative interactions, exploring an approach based on a conflict model.

We start this review by addressing the topic of Conflict Theory, describing the background for our conflict model approach.

The expression of personality and believability are two intrinsically connected topics. In fact, we arrive at our goal of creating an agent that expresses personality as a means to achieve believability. This way, we describe the topic of believability to provide further context, which relates to personality, the focus of our work.

We present some examples of previous research related to our work, and have a section referring some existing systems of cooperative agents, in particular.

With the intent of creating a system to be used in a computer game, we also present the alternatives of agent architectures used in the games industry.

2.1 Conflict Theory

Throughout our lives it is inevitable to encounter a conflict situation. While it can lead to negative effects, like destroying social bonds or provoking violence, it is believed to have also the potential for social change and evolution [5].

In a conflict situation, individuals are faced with the challenge to overcome their differences of interests, which, for better or worse, can lead to new interesting situations, depending on the resolution strategies used by each party. Conflict is even compared to a play or novel by Laursen and Hafen [6], who state that “There is a protagonist and an antagonist (conflict participants), a theme (conflict topic), a complication (initial opposition), rising action (conflict behaviors), climax or crisis (conflict resolution), and denouement (conflict outcome)”.

6
2.1.1 Conflict Definition

There are different views on the definition of social conflict, although all closely related. Kenneth Thomas stated in 1992, that there were two main perspectives on conflict definition: one perspective relating conflict to deliberate interfering with the goals and intentions of others; the other perspective includes as well the events, like the parties’ perceptions, that lead to the conflict process. As mentioned before, Thomas’ own definition states that conflict is “the process which begins when one party perceives that another has frustrated, or is about to frustrate, some concern of his” [4].

Morton Deutsch provides a general definition of conflict that relates to any situation where the involved parties have incompatible activities, such as opposing goals, wishes, feelings or values [7]. These perspectives on conflict are useful for our work, where we deal with situations where there are possible interferences and incompatibilities between the player and the agent’s goals.

2.1.2 Conflict Models

Researchers have evolved different conflict handling models. Coleman et al. [5] describe some of the most relevant models for conflict resolution:

- Deutsch’s Social Interdependence Theory [7], explores the types of interdependence of goals (positive or negative) between the party, comparing effects of cooperation versus competition in conflict. This theory has its limitations, such as assuming high degrees of interdependence between the parties in conflict.

- The Social Motivation Theory [8] compares the different social motives of the involved parties, focusing primarily in pro-self versus pro-social motives, related respectively to the pursuit of individualistic and competitive goals, versus a more altruistic and cooperative behavior.

- With the Dual-Concern Theory [4], Kenneth Thomas provided a model that extends Deutsch’s competition-cooperation model, into a two-dimensional model that expresses concern for own goals and concern for the other party’s goals as orthogonal interests.

This last model, while conceptually simple, provides a strong basis for characterization of individuals’ conflict strategies, and it was chosen as a foundation to build different parameterizations of the game companion agent. This model describes five conflict-resolution strategies, or conflict-handling modes, defined by the combination of values of assertiveness (concern for self) and cooperativeness (concern for the other). These different conflict-handling modes, which we address in the next section, show great potential be directly linked to a personality style.

2.1.3 Conflict-handling Modes

Believing that each conflict handling mode can be related to a personality type, we chose to use Thomas’ conflict model to characterize the agents’ behavior choices.
This model consists on classifying conflict resolution according to two orthogonal axis: *assertiveness* and *cooperativeness*. While *assertiveness* describes the interest in satisfying one’s own goals, *cooperativeness* describes the interest for the other party’s goals. Different positions on this dual-concern diagram describe different conflict handling modes. This model defines five handling modes: *competing*, *collaborating*, *compromising*, *avoiding*, and *accommodating*. [9]

The *Competing* strategy, where the individual has maximum interest in fulfilling his own goals with no regard for the other party’s (high assertiveness and low cooperativeness), has been associated with forcing behavior and win-lose arguing.

As an opposite strategy lies the *Accommodating* mode, where the concern for one’s own goals is minimal and while concern for the other party is high, being unassertive and very cooperative. This strategy searches to accomplish harmony, soothing the other person.

The *Avoiding* mode, characterized by no interest in pursuing either one’s own nor the other’s goals, with little assertiveness or cooperativeness, is associated with evading the conflict situation and not succeeding in taking a stand.

With the *Collaborating* style there is high assertiveness and cooperativeness, displaying high concern for both the involved individuals’ goals. Individuals employing this strategy seek to confront their incompatibilities and achieve a solution through problem solving.

In *Compromising* mode lies a mid-way style between Avoiding and Collaborating, with medium concern for either party’s goals, having average values of assertiveness and cooperativeness. In this strategy the individual seeks a middle-ground position to handle the conflict.

Although this model focuses on describing conflict handling modes by the two dimensions of cooperativeness and assertiveness, Thomas notes that the five modes can also be described in terms of *integrative* and *distributive* dimensions.

Here, the integrative dimension lies in the oblique line joining the opposite modes of *Avoiding* and *Collaborating*, and represents the total of satisfaction that both involved parties will get, relative to the
individual's behavior and investment in the conflict resolution.

The distributive dimension is described by the oblique line running from Competing to Accommodating, and represents the distribution of satisfaction of goals that will be given to each involved individual.

This way we can describe Competing as the taking strategy and Accommodating as the giving strategy (with the other three modes being intermediate) in regards to the distributive dimension. Following the integrative dimension we can relate the Collaborating strategy to maximizing the resulting satisfaction of the whole party's goals, and Avoiding to minimizing it by neglection, while the rest of the modes are intermediate [9].

This information description reinforces the value of the model and shows great improvement over simpler models like Deutsch's one-dimensional cooperative-competitive model.

2.1.4 Conflict in Games

Conflict is a research topic which has been applied in the creation of computer games, albeit mostly to serious game environments with educational purposes.

An example of such projects using conflict theory, and Thomas' model in particular, is the work of Cheong et al. [10]. In this project the authors present a Computational Model of Conflict and Conflict Resolution, consisting on five different stages: (i) conflict situation creation, (ii) conflict detection, (iii) player modeling and conflict strategy prediction, (iv) conflict management, and (v) conflict resolution. This system was applied to a Resource Management Game, where the conflict is generated through the limitation of resources. In this environment, the player must resolve the distribution of limited resources, and is incentivized to reach a fair solution for the distribution.

Another project where conflict is applied to a game is FearNot! [11], a serious game where the narrative outcomes are generated as a result of the actions of a character advised by the player. This system focuses on creating a believable environment where children (the players) are able to explore different conflict resolution strategies to handle the conflict situation, which is bullying. This system does not however employ an explicit model of conflict.

Finally, another example of a project that relates to our problem is "My Dream Theatre" [12], a serious game developed with the purpose of teaching conflict resolution to children. In this game the player plays as the director of a school theatre club who must manage conflicts of interests between the theatre’s actors. This project uses Thomas' model for conflict handling modes as a way to define the conflict resolution styles of the actor NPCs (Non-Playable Characters), who exhibit different preferences of interaction accordingly.

2.2 Believability

As stated by Bates [13], a believable character must provide the illusion of life allowing the audience's suspension of disbelief, so they feel that the agent is real. This way, in order to design a believable game agent, we must provide the illusion that it has its own desires and feelings about the world. When
a character shows no expression of emotion or care about the surrounding environment, the audience
(which in a game corresponds to the players) will not feel any empathy towards it, and therefore not care
for it either. In the words of Bates “the emotionless character is lifeless as a machine”, and a believable
agent is an autonomous agent that has specific rich personalities, like a character in a movie [13].

While a believable agent is not required to be very realistic, it is important that it should effectively
express its personality and role in the world [14].

A prominent work in this context of designing believable virtual characters is Bates’ Oz project [13]. In
this work, a virtual world was created including different characters, named Woggles, which follow a goal-
directed behavior-based architecture for actions, coupled with components for personality expression
and emotions capabilities.

Another example of a project focused on creating believable characters for an interactive environment
is the work of Magerko and Laird et al. [15], that implemented an architecture for developing complex
AI characters, in the context of computer games, embedding AI agents as actors and directors in a
narrative setting.

Another meaning of believability for agents in games can be “giving the feeling of being controlled by
a player” [16]. Systems like the work of Tencé et al. [16] and Le Hy et al. [17] try to imitate human-like
behavior to create the illusion of playing with a person, so that the agents display believable human
behaviors in the games.

However, the purpose of our work relates to designing a believable companion character that belongs
to the game world and not the attempting to emulate human players, so we focus on the meaning of
believability related to the illusion of life.

According to Bates’ Oz group, there are some requirements that an agent must fulfill in order to be
believable [18]:

- Personality - An agent should exhibit personality in their every action, from animations to behavior
  choice, making the character more unique and specific and therefore more interesting.
- Emotion - According to their own personality, the characters should express their emotions.
- Self-motivation – Agents should have their own desires and pursue them, being more than just
  reactive entities.
- Change – Agents should show change and growth over time, in a way that is consistent with their
  personality.
- Social relationships – Through interaction, agents should evolve relationships with others.
- Illusion of life – Providing the viewer with the illusion that the agent is alive requires that it can
  exhibit capabilities like movement and language, and react to the environment and act in the world
  pursuing simultaneous goals and actions.

The importance of imbuing virtual agents with personality and emotion is also emphasized by Gratch
et al., who state that, since people express their emotions and personality through verbal and non-verbal
behavior, it is essential to model these features in the agents’ behavior in order to achieve believability [19].

We conclude that personality and emotion are two main aspects on creating believable agents, and a considerable amount of research on believability focuses around these two topics.

2.2.1 Personality

As mentioned before, exhibiting personality is paramount for believable agents. Personality relates to the motivations of behavior and regulates how and which goals are pursued [1]. In order to be believable, an agent is not required to choose the best plans and actions, and have the best possible performance in the game, but it is imperative that its every action shows its own style and personality [18].

There are different approaches to define and characterize personality types. Two relevant systems for evaluating personality are the Five Factor Model and the Myers-Briggs model.

The Five Factor Model [20] is based on the Big Five personality traits, describing five different dimensions of personality: Extroversion, Agreeableness, Conscientiousness, Neuroticism and Openness to Experience. This model presents a hierarchical structure and each factor is a broad domain that has its own subset of correlated traits or scales which better define the personality dimension. For example, when classifying the Neuroticism dimension there’s the traits of Anxiety, Hostility, Depression, Self-Consciousness, Impulsiveness and Vulnerability.

Another well-known attempt to analyze and classify personality types is the Myers-Briggs model [21]. This model describes four dichotomies: Sensing-Intuition, Thinking-Feeling, Judging-Perceiving, and Introversion-Extroversion. The model states that for each dichotomy, none of the two different psychological types are better or worse, but each person has a preference for one type. This way, individuals are characterized by the combination of these type preferences for the four dichotomies.

Since with this work we intend to create agents able to express personality through different behaviors according to the Dual-Concern conflict model, it is interesting to take these personality models into account and note that there has been research on correlating the conflict handling modes to these personality theories.

Of particular interest, Thomas and Kilmann [9] have studied how the Jungian personalities of the Myers-Briggs model relate to the different conflict management modes. While the Sensation-Intuition and Judging-Perceiving dimensions do not appear to relate to specific choices of conflict handling modes, stronger correlations were found for the other two dichotomies. In the study a model is proposed mapping the introversion-extroversion and feeling-thinking dimensions to the integrative dimension and distributive dimension of conflict behavior (described in the Conflict-Handling Modes Section) respectively. This work suggests that there can be a relation between conflict handling modes and personality types, which supports the base idea for our work, to use the conflict handling modes to produce expression of different personalities. In particular, we attempt to express the close relation between the integrative dimension of the conflict model and the introversion-extroversion dimension, as a result of the character will to take action.
The Five Factor Model was used by Bono et al. [22] who also studied the relation between personality and conflict between two individuals. This study once more suggests our base idea that conflict and personality can be correlated, linking personality traits to tendencies for different conflict experiences. While this work shows an interesting direction to relating conflict and personality, for our approach we chose to take a more simplified direction, to facilitate easily parameterizing the different variable personalities.

An example of research on designing believable characters with personality is the work of Rizzo et al. [1]. In this work, an agent with a model for personality integrated in a plan-based architecture is implemented, where behavior characteristics vary according to different personality types. The work focuses on the personality dimension of “help-giving”, portraying characters in the selfish to altruistic spectrum. Here, the different personality types influence behavior by regulating the resulting choice of goals and plans pursued by the agent. For example an altruistic character will have preference for resource provision. The problem addressed by this work is very close to ours, but focusing on a different dimension of personality. The close similarities make this a very useful resource, and this project serves as inspiration for our work, as we also use this idea of assigning preferences for actions, giving priorities for certain behaviors depending on the personality of the agent.

The GOLEM project [23] also approached to topic of agents with personality. Their model is applied in a multi-agent cooperation context, and focuses on the activities of “delegation” and “help”. They define the agents’ personalities as a combination of traits related to those two activities, and employ a rule-based system for determining which actions to take according to the personality. While this project bears some similarities to the goal of this thesis, being related to personality in a cooperative context and dealing with resolving conflict situations, the rule-based approach regulated by a very limited set of traits is quite limiting, and we try to supply a more flexible solution allowing for more interesting and varied interactions.

2.2.2 Emotion

The other element we defined as a critical aspect to take into consideration when designing believable agents is the expression of emotion. As in real life a person’s emotions have an impact on decision-making and actions, a believable virtual agent should also exhibit different behaviors expressing an internal emotional state [24]. There is a strong close relationship between emotion and personality. Personalities are usually described by a different mapping of goals, emotions and reactions [1]. This way, different emotions are triggered by reacting to the environment or state of a goal, depending on the type of personality.

According to Gratch et al. [19], research focused on exploring emotion to implement believable agents can be divided in two approaches: communication-driven and simulation-driven. While the first one focuses on deliberately conveying emotions based on what is the desired resulting emotional reaction of the user, and so must take into account the perspective of others, the second approach focuses on simulating real emotions, using models for appraisal of emotion. The most prominent theory for emotion appraisal, when applied to computational systems, is Ortony, Clore and Collins’ cognitive appraisal
theory, commonly referred to as the OCC model [19]. In this model, emotions arise from reasoning about events, agents and objects, and represent valenced reactions to these perceptions of the world. Regarding events, one may be pleased or displeased with the consequences, for oneself or the others. For agents, one can approve or disapprove of their actions, others’ or his own. For objects, one may like or dislike their aspects. These different valenced reactions when evaluated will then lead to emotions [25].

A large amount of research on designing believable agents with emotion uses the OCC model. The work of Marsella and Gratch is an example of projects using the OCC model, having implemented a plan-based virtual agent model with emotion appraisal based on OCC theory [24], and in a later project combined the OCC with coping theories analyzing how people cope with strong emotions (see [19]). The previously mentioned FearNot! project [11] employs this model for emotional appraisal, using the emotion-driven architecture FAtiMA [26]. The Fatima model uses OCC theory for the agent’s emotional appraisal and extends its emotional proficiency to incorporate emotional deliberation, including a memory module giving the potential for the agent to also reason about recent events [27]. This model has also been used as base for other projects researching emotion and believability for virtual agents, such as the work of Lim et al. [28], which explores a new implementation of memory for virtual social companions.

From the aforementioned works we can notice the importance of emotion as a social aspect of interaction. It is useful to look at how the OCC theory is used for the implementation of emotional appraisal in these projects, leading to different emotional states and influencing the agent’s choice of behaviors and gestures.

Since the focus of our work is personality and due to time constraints we did not address in depth the emotion expression view in our system. However this is an interesting topic that is worth future investigation as an extension to our work. It would be of interest in the future to explore the ideas expressed from the above mentioned studies, and attempt to increase the believability of our AI model, complementing the personality aspect with the also very important aspect of emotion expression.

2.3 Cooperative Agents

In games where the player interacts with virtual characters working towards common goals, the need for satisfying cooperation interactions requires additional concerns when designing the cooperative agents. In this type of setting, agents need to infer the tasks players are pursuing and respond with an appropriate behavior taking into account the context and environment [29].

2.3.1 Cooperative Agents in Research

A considerable amount of research has addressed the topic of cooperation with virtual game agents in the context of team-mates, as presented by McGee and Abraham’s review article [30]. However, most of these works focus on larger teams of multiple agents interacting with the player and with each other. With our work we are more interested in research related to partner companions with expression of
personality, in cooperative game scenarios.

To facilitate the development of cooperative characters, Denzinger and Winder [31] propose a simple framework for building intelligent cooperative agents through evolutionary training and coaching. This work tries to automatically configure agents for best performance within the game using genetic algorithms choosing the fittest. They add the concept of coaching as a way for developers to accelerate the evolution of the agents by supplying initial strategies, providing key desirable situations. This work, however, focuses on improving the performance of the cooperative agent, disregarding the issue of believability, which is of utmost importance for an immersive game experience. Evolving initial parameters is however an idea we would like to explore, to extend our initial mapping of personality to actions.

Exploring the problem of expressing personality in teamwork games, Tan and Cheng [32] focus on adaptation of game agents, using a module that assigns values to the agents’ actions depending on their personality. While they do not mention how personality is assigned to an agent, they implement it as functions that assign values to actions describing the chance for each action to be selected. This project seeks to create a framework for an agent that will automatically conform to the style of the player, so they perform better as a team. This adaptation is executed between gameplays, by reinforcement learning, updating weights of the adapter’s module neural network, which will change the initial personality towards one that best fits the player. The approach taken on this work shows an interesting perspective for designing a believable cooperative agent with adjustable behavior capabilities. The random factor when selecting actions should provide more behavior variability, and we used this idea in our model, adjusting the behavior choices with a randomized deviation in order to produce behavior choice variability. Although the process of assigning personality is not mentioned, the approach taken by this work is similar to ours, as we also implemented personality parameterized by a set of functions of utility that assign preferences to actions. The reinforcement learning described by this work is also of interest for extending the base model, presenting the possibility of fine-tuning the initial parameters, mapping personality to a behavior choice, from gameplay execution.

2.3.2 Cooperative Agents in the Games Industry

In this section, we present some examples of commercial games worth mentioning in the context of designing believable partner characters.

Valve’s Left 4 Dead series[^1][^2] is a good example of satisfying interactions in a cooperative setting. Taking place in the aftermath of a pandemic outbreak, the game encourages cooperative play between player and team-mate agents (although multiplayer play is also possible) with the purpose of surviving hordes of zombie-like creatures while progressing through the narrative. There is great depth to the aspect of cooperation in this game, with a great part of the game design focusing on this mechanic. The team of survivor agents display great cooperative interactions with the rest of the agents and player, serving as inspiration for creating good cooperation situations for the environment for testing our agent. The agents in the team also present high believability, each one having its own identity, which relates to

[^1]: www.valvesoftware.com
[^2]: www.L4D.com
our goal of imbuing a character with personality, and emitting various vocalizations, pointing useful information or calling for help. The fact that the agents display helplessness and share the same strengths and resources as the player also adds to the feeling of being human and believable.

In the more recent game Bioshock Infinite we can see an excellent example of believable virtual agent design taking the role of the companion character Elizabeth. Elizabeth accompanies the player throughout most of the narrative and does more than just providing occasional help. In fact, the behavior of Elizabeth is very well characterized, portraying the personality of a fragile girl, hiding, and only supplying the occasional useful item, when danger arises. The AI companion strongly expresses emotions, exhibiting not only emotions like fear, but also marvel at the surrounding environment, through verbal behavior and facial expressions, as well as through body expression and actions. The Elizabeth AI excels at giving the illusion of real life, with real desires, emotions and personality, and also presents very solid interactions with the player, even not blocking paths. For these reasons, this character is a great inspiration for our work, and we would like to create an agent that shows the personality expression capabilities of Elizabeth, with appropriate expressions and actions, and with the additional parameterization for varying personalities.

2.4 AI Models in the Game Industry

In order to create interesting AI characters, there are varied alternative solutions currently used in the game industry.

Finite-State Machines (FSM) are very commonly used for behavior modeling. Using this architecture, a character’s behavior is broken down into states, such as “Idle”, “Chasing” or “Fleeing”. Only one state may be active at a time, and the agent switches between states through transitions, usually associated with specific conditions that lead to the transition [33], for example being threatened could lead from an “Idle” state to “Fleeing”. Although initially simple and easy to visualize the logic, this model begins to explode in complexity when many states are introduced, making it confusing and error-prone.

This problem is somewhat lessened by Hierarchical Finite-State Machines, where, the states can be themselves other basic FSM’s [34]. By introducing a hierarchy where a state can contain other states some state duplication is no longer required. For example a hypothetical state of “Guarding Building” could be made of two sub-states like “Patrolling to Door” and “Patrolling to Safe”, and if we would like to be able to transition from both these states to a “Conversation” one we could just define the transition from the higher state “Guarding Building” instead of having to define multiple transitions and conversation states specific to each of the existing patrolling states [35]. This hierarchical model makes the graph of states and transitions more clear and provides better structural control by breaking down complex states. However, there is still the problem of complexity when there are a lot of transitions between states, and in that situation other algorithms are more adequate.

Goal-oriented planners, provide an architecture based on collections of hierarchical goals [33]. Each goal can be atomic, defining a single action, or composite, describing more complex tasks, comprising

---

3 www.bioshockinfinite.com
other subgoals (for example, going to the movies could require driving to the movies, buying ticket and watching movie). Actions are associated with preconditions and effects, and each desired goal is pursued by working out which actions must be completed to achieve it, and assuring that their conditions are met, successively. Goals may be reached from the effects of different actions, so the agent may perform different plans (sequences of actions) to achieve a same goal.

In order to achieve a goal, the planner agent will successively find the actions whose pre-conditions can be met and fulfill the required conditions of the successive subgoals and planned actions, generating a plan of execution. This architecture makes designing the behavior simple, by focusing on simple components, and makes it possible for agents to choose their own behaviors, based on the available actions, but this can also give unusual results. Since the agent can plan any sequences of actions that satisfy the goal conditions, unanticipated awkward plans may be assembled [35].

Behavior trees have become a popular structure for creating AI characters. Very similar to Hierarchical State Machines, Behavior Trees describe behaviors by a hierarchy of tasks, instead of states [36]. The kinds of tasks that compose a behavior tree are Conditions, Actions and Composites. Conditions are associated with Actions, establishing tests that must be checked in order to successfully execute an Action. Conditions and Actions comprise the leaf nodes, while Composites make up the branches. Composites are usually either Selectors or Sequences. While Selectors encompass alternative actions, and the first possible one is selected, Sequences hold a list of actions to be executed in succession. The algorithm execution progresses successively choosing for each level of the tree which behavior to execute based on the preconditions for the tasks.

This is a very simple architecture, and behaviors (as sets of tasks) can be designed independently of each other, making it easy to add or remove behaviors from the tree. While the base algorithm does not take memory into account, and behaviors are considered exclusive, this architecture makes it easy to extend the algorithm to take these into account. Behavior trees seem to be the most adequate architecture for our solution, being flexible, intuitive and widely known and used, and were therefore the choice of architecture for implementing our agent's behavior.

Utility-based systems are an alternative to systems based on simple predicates, where actions are chosen when their predicate conditions are met. Using utility theory, the decision making process, for choosing which actions and behaviors to perform, is conducted based on each action's single, uniform value, that describes that action within a given context [37]. These values, referred to as utility, illustrate the usefulness of performing the action in the considered background. The functions that are used to determine the utility values usually take as input elements such as environmental factors and the agent's state. The expressions can take many forms, from linear combinations of values to non-linear expressions like quadratic curves, and can even be defined as piecewise curves. The utility scores must be consistent, so normalized values are usually used to be compared, and to a reach a decision on which action would be more useful. This analysis can then lead to the selection of the more suitable actions when planning the agent's behavior.

Although the core framework for utility-based systems is simple, and it is easy to add different results, it can be hard to tune in order to have the right behaviors activated [35], but this theory is useful for our
work, and we used this utility-based framework combined with the underlying behavior tree structure in order to produce better results.

2.5 Concluding Remarks

In this chapter we overviewed the state of the art relating to the problem of creating a believable AI companion by the manifestation of different personalities based on a conflict handling model. We conclude that when creating a believable character agent, it is of utmost importance to successfully exhibit personality and emotion. Of particular interest we see personality expression as a way to significantly increase any AI character’s believability, and see it is a challenge that many studies approach in different ways.

We see situations arising from a conflict of interests as an appropriate means of expressing different personalities’ choices. Inspired by some studies described in this section, done in a similar context, we base our work in Kenneth Thomas’ definition of conflict, and establish as the foundation for our model Thomas’ conflict handling model. Like in the projects cited above that use this model of conflict, our AI character will exhibit different preferences according to the the assigned conflict handling style. The works described in this section present some interesting ideas, some of which we chose to include in our own approach. From the work explored, a couple of aspects we were inspired to include in our approach were: the introduction of a random factor to increase variability of chosen behaviors, and parameterization of behavior choices according to utility functions.

From the work explored related to conflict in games, we are challenged to take a different approach attempting to apply it as a means to create the foundation of an AI personality and use it in a non-serious game environment, attempting to provide the games industry with a new approach to expressing personality and therefore help improving the developed characters believability. From the techniques researched for creating game characters agents, we chose to base our work on the Behavior Tree architecture, which appears to be the most appropriate for our target environment, and extend this base model to include dynamic Utility evaluation.
Chapter 3

Approach and Architecture

As previously mentioned, in this work we propose to implement an agent model capable of expressing different personalities depending on the assigned conflict-handling mode.

We intend to create an interactive experience where the player will interact with the bot in first person, and have the chance to see the manifestation of the bot's personality via the real-time choices it will make throughout the level, in an environment experience as if playing with a partner, trying to emulate the experience of playing with another person, which should verify the believability of the bot and the personality model.

In order to achieve this, we propose a model using a Behavior Tree infrastructure and employing Utility Evaluation module for assigning preferences to behaviors.

We implemented a game in UDK, creating a testing environment for our agent, which we then used for user testing, in order to evaluate our solution.

All these issues, which define our proposed solution model, will be further addressed in the following sections.

3.1 Experience Design

To provide the player an experience of interacting with an agent with a believable noticeable personality, we designed a scenario experience that provides enough situations to allow the bot to visibly manifest specific traits through its behavior choices commonly associated with the personality type.

We chose to create an FPS style game scenario where the player would interact in first person with the bot and the environment, in a fast-paced experience, with the final challenge of finding a way to escape the building.

Having in mind a model for personality based on conflict handling modes, it is important to design a level experience where the team of bot and player have to deal with some conflict of interests, leading them to choose how to handle the conflict, and therefore manifesting their personalities. For example, having a situation with few resources available would manifest the concern for well being of the team or self interests. In particular, a very Cooperative bot would manifest its concern for the player by leaving
Having chosen to design a companion bot, we should create experiences where the bot and player buddy-up to progress through the level. This way, we aim to create challenges that could only be overcome by cooperation of the bot-player team. An interesting combination for this challenges is to create a variant where not only do they need to cooperate, but also the outcome would only benefit one of them.

Taking into account the intended interactive environment/experience, and the cooperation and conflict challenges that will be present in the level/environment, the following behaviors are expected from each style of conflict management:

**Avoiding:**

- agent sometimes performs only easy, irrelevant/low-consequential tasks
- will not care for the player’s concerns, nor its own
- will not fight for hard objectives
- won’t make efforts to help the team’s objectives

**Competition:**

- agent will disregard player’s intentions
- will follow its own agenda, trying to fulfill its own objectives ahead of the team’s objectives
- will keep advantages for itself (such as ammo, goods, good position)
- will put its safety ahead of the player’s
– will not follow directions or requests from the player

**Accommodation:**

– agent will try its best to fulfill the team’s objectives
– will put player’s safety ahead of his
– will give the player all the ammo and goods
– will follow any instruction the player gives

**Collaboration:**

– agent will follow directions and requests from the player
– will try its best to fulfill the team’s objectives (actively try to pursue team goals)
– both agent’s and player’s objectives are equally important (but team goals are more)
– will give items to the player when the player needs them and the agent has spare

### 3.2 Agent Architecture

In order to build an agent to behave in the intended ways as outlined by the scenarios in the previous section, we designed our Agent Model with a structure as depicted in Figure 3.2.

![Figure 3.2: Agent Model Process.](image)

The **Agent Controller** is the main structure that controls the Bot’s behavior and communicates with the external elements, such as the Agent Pawn (that represents its presence in the world), and the World environment.

The main component of this structure is the **Behavior Tree** module. This module acts as the decision process for the Controller, interfacing with it by providing the decision outcomes that the Controller can act on and pursue. The Controller also communicates with the Behavior Tree in order to feed the context variables that affect the decision process, which are stored in the **World Info** and **Personality** structures. At certain key situations (such as getting shot or finding a new important scenario objective),
the Controller can also request a reordering of the Behavior Tree, so that the behavior choices evaluation reflects the new context more dramatically.

The Behavior Tree module includes an Utility Evaluation function that will guide the path taken in the Behavior Tree evaluation, both by yielding which Tasks are pursuable at each time and context, as well as the defining the sorting of Tree when a reordering is called for.

We designed our solution seeking to maximize flexibility and reusability of its elements, in a way that we could easily swap a submodule with one with a different implementation as long as it keeps the defined interface between components. For example, we could use a non-humanoid Bot companion Controller, who would, for the same Behavior Tree, manifest its actions in the appropriate non-humanoid way (like biting instead of shooting a weapon, when the chosen behavior is “attack”).

The tasks that were planned to be included in our Behavior Tree mostly consist of exploring interactions with the player, regarding the available supplies and plan enforcement possibilities, and combat actions for shooting, hiding in cover, and other typical FPS actions. This model could easily be applied to another game environment, by composing the Tree with other kinds of Tasks.

In the next sections we describe our model in further detail.

### 3.2.1 Behavior Tree

As stated before, a Behavior Tree is composed of Tasks, which can be Actions, Composites or Conditions. In our particular implementation of this model, we include in each task a built-in precondition verification step parameterized by the environment context (as perceived by the agent) and the personality values. The combination of these values is used to calculate the task's Utility function, and it is generally this utility value that will be used to evaluate if the preconditions to pursue the task are met. This precondition verification mostly consists in defining a threshold for these values that affect the task’s viability: if the evaluation results are above the threshold then the task is considered viable to pursue. Let's consider the task “Treat Wounded”: a Utility for this task is evaluated by combining the factors of how much health is missing from each member of the team, and how much does the Bot care for each member of the team. If this combined value is above the defined minimum interest, then it will pursue this behavior. For some tasks, such as this one, the value for the minimum interest threshold is also parameterized, in this case by the Bot's personality values for Assertiveness and Cooperative (the smaller the sum of these values, the less the Bot cares for healing the team, and therefore the threshold is higher). For some tasks, besides the utility threshold evaluation, some other preconditions can be verified: for example there is no point of pursuing a “Battle” task if no enemy is around.

As the tasks can be not only unit Actions but also Composites, in the latter case the evaluation will have to pass the successive children's utility and threshold analysis in order to follow the most appropriate child task.

The main Behavior Tasks that compose our Behavior Tree consist in Treat Wounded, Battle (Hide and Fight), Puzzle Solving, Mission Objective Solving, Exploration. The overall Behavior Tree we implemented is later depicted in Figure 4.4)
3.2.2 Utility Evaluation

As mentioned in the previous section, each task has an Utility function that depends on the world state and personality. This utility is used not only to check the value to match against thresholds (whose values are parameterized for each task, can depend on the world state and personality as well, or just be static), but also for establishing an ordering between tasks on the same tier of the tree which we use for reordering the tree at important moments in the flow of events (as we will further describe in the subsection 3.2.3).

The utility evaluation of each action will be regulated by the position in the conflict model’s two dimensions, of cooperativeness and assertiveness, which was assigned to the character on its creation. The utility module will comprise a list of utility functions for each action of each behavior. These functions take as parameters the values for cooperativeness, \( c \), and assertiveness, \( a \), from the coordinates \((c, a)\) taken from the position in the model. Additionally, these evaluating functions take into account environmental factors of the surrounding world, the agent’s state, and the state of the player. For both the agent’s and player’s states, besides the “physical” characteristics such as healthpoints and ammunition, their internal goals (which, for the player, is information that is inferred from the player model) are also considered. These goals are represented as a state that must be reached, for example “enemies vanquished”, “position reached” or “item collected”.

As an example, let’s consider the action “take item”: The utility function regulating the preference for this action should have as a positive factor the assertiveness value \( a \). This way, an increasing value of \( a \) will increase the preference for this action, favoring a behavior more characteristic of the Competing strategy (which also goes in line with the previously mentioned distributive dimension).

On the other hand, for an action such as “give item”, the utility function should take the cooperativeness value \( c \) as the positive factor, so that the higher values of \( c \) will favor a behavior closer to the Accommodating strategy.

Taking all the factors into account, which should be calibrated by coefficients, each utility function should represent the combination of relevant factors for the action.

\[
U_{\text{action}}(c, a) = f(c, a, F) \tag{3.1}
\]

where \( F \) is the vector of all the factors, \( f_1 \) to \( f_n \), that affect the utility of the action.

Since a behavior comprises a set of actions, the preference for each behavior will ultimately be the result of the combination of the evaluated utility for every action in the behavior, since at each decision point in the tree, the preference for which task and sub-task to expand will depend on the successive utility evaluation of the children tasks.

This way, we make the appropriate behaviors more desirable, by parameterizing the module in a way that the agent will consistently choose appropriate behaviors, matching the assigned conflict interaction style. These personality-consistent and contrasting behavior preferences should give the player the right perception of different personalities.

Some of the factors that are considered by the utility evaluator are: Own Health, Player Health,
Ammo, Health Kits, Number of Nearby Enemies, Enemies in Line of Sight, Distance to Enemies, Distance to Health Kit, Distance to Cover, Distance to Personal Objective, Distance to Team Objective. All this information, including each team member's personal inventory, is accessible to all the team members, being displayed visually to the player. These factors should be relevant for conflict situations in a cooperative context. The quantity of Ammo and Health Kits, for example, should provide a situation where a choice must be made on how to deal with limited availability of these items. The agent will exhibit different preferences on its interaction and cooperation with the player in this situation, by either favoring assertiveness, and keep items to itself (or even request more from the player), or cooperativeness, and try to satisfy the player's needs offering the items.

Another significant situation can be created from the arising danger when there are enemies in line of sight. Depending on the agent's inclination regarding cooperativeness and assertiveness, very different behavior choices can be made. A higher value of assertiveness with low cooperativeness will likely lead the agent to put its own protection above the player's, taking cover or leaving the player behind, while a lower value of assertiveness and high cooperativeness could produce a more protective behavior. With both cooperativeness and assertiveness high values, the agent will have a tendency to actively try to defeat the enemies, balancing its own safety as well as the player's.

Special attention must be given to defining appropriate utility functions, since this is what has a greater impact on the manifestation of behavior choices, which need to be realistic, meaningfully showing the intended personality choices for the situations described by the scenarios. This way, the utility functions must be improved iteratively over many runs of the scenarios for the varying personalities. To make this process easier, we took special care in our implementation to make every task highly parameterized, without built-in values, and therefore allowing for an automation of this procedure by some sort of learning mechanism, in future work.

By adequately parameterizing the action utility evaluator, each type of character should present quite different, yet consistent, behaviors that match the perceived personality.

### 3.2.3 Reordering of the Tree

While different situations can lead to different utility thresholds for the behavior tree's tasks, and therefore lead to different choices of behavior construction, the issue with behavior trees is that they remain static and do not adapt themselves. We thought that key moments could present a bigger impact on the behavior choice, switching the order of evaluation of the tree's tasks. This way we use the utility evaluation not only to determine which branch to pursue but also in special occasions lead to a new ordering of tasks from which to chose what to pursue. For example, let's say the Bot was currently engaged on a task of solving a puzzle and sees a new enemy. Instead of simply carrying on the task of solving the puzzle, we believe that at this key moment the tree evaluation should be restarted, and new priorities defined, in a way that the impact of that added menace to the team can be emphasized such that it becomes the first thing that is evaluated.

The Behavior Tree is a flexible structure, where we can swap many of its nodes, for example changing
the order of selector tasks’ children, creating a new flow of the decision process. However some tasks swapping could yield illogical behaviors, so we chose to restrict task-reordering only to nodes within each tier of the tree, and in a way that would not break sequences that should not be split. We apply this restriction by having the parent tasks be responsible for reordering their children, and if some children should not be reordered, then the reorder function of the parent should reflect this, in a case by case situation.

Even with some restrictions to some children tasks, task-reordering can possibly lead to not very rational behaviors. Since we use the Utility function values for each task to define the outcome of the reordering, it is of outmost importance to ensure that these Utility functions are well-parameterized and well-adjusted, as well as and properly normalized, so that the resulting ordering is the most appropriate for the environment context. Failure to do so would make it that the Bot could be perceived as irrational, or unhuman-like, and would most likely break the believability we strive to achieve, causing some detachment of the player.

3.2.4 Personality and World Perception

The most important factors that should influence the agent's behavior are its perception of the world and its personality. We designed our solution in a way that the Behavior Tree and Utility modules have access to these values, via the agent Controller who holds this information in its internal blackboard. While the World Information values change dynamically over the level progression, the Personality values are final from the moment they are setup. These personality values are fed to the agent setup externally, allowing to easily create different personalities.

In order to update its perception of the World, alongside the Behavior Tree task choice evaluation, the Bot Controller periodically analyses its surroundings and updates the blackboard. Additionally to this periodic check, specific events will trigger an update of the Bot's perception (e.g. a new enemy appears, the player issues a new command). If an important change is noticed, then this would trigger the reordering of the tree.

Concluding Remarks In this chapter we presented the base model architecture that we used to implement our companion Bot. The agent model is based on a Behavior Tree architecture, evaluating the tasks it pursues to express different behaviors according to the world context, its own state and its personality values. The combination of these values is used to calculate the utility of each of the tasks, which is used as the precondition verification for each task, as well as for determining the reordering of tasks, that is triggered at key situations that have the potential to greatly impact the Bot.

In the next chapter we describe the actual implementation of this model.
Chapter 4

Developed System

As a project targeted at a game environment, the chosen platform where to implement our solution was Unreal Development Kit (UDK). This platform provided some helpful tools, and we could make use of its underlying architecture for character controllers (the Artificial Intelligence components) to build our Agent Model upon.

The testbed scenario also benefited from the tools already provided in UDK, and we could use and extend many of its sample assets to populate our world.

We managed to make use of the Toolkit provided by UDK, and implemented connections between the code and the design tools provided. As a result of these efforts, we created for this environment a framework for implementing Behavior Tree-based agents, highly extensible and easily customizable via UDK's visual programming tool, Kismet.

In the next sections we describe in further detail how we implemented the many components to our system.

4.1 Unreal Development Toolkit

The UDK tools that were of particular interest to our project were the Editor, Kismet and Matinee.

The Unreal Editor is the environment where we design and build the physical aspect to the virtual worlds that make up our levels. Here we can place the objects and actors we want to interact with, that we created by code in Unrealscript.

Kismet provides a visual environment to program game logic that makes it easier for game designers who have no actual programming experience to compose and modify the flow of the game. We created some modules for Kismet, taking the form of logic boxes that can be connected and translate to their code equivalents. For example, the creation and setup of our Bot was managed via Kismet, using a custom Personality setup box and the existing Actor-Spawning modules. The Kismet sequence for the Bot initialization is shown in Figure 4.1.

Matinee is a tool for key-framing the properties of Actors in the scene over time, which allows to create not only animations but also cutscenes. To our project this tool was of particular interest to create
extra animated actions, such as our puzzles activation results and the doors opening mechanism.

The base implementation of the system, however, lies on what was done in scripting using Unrealscript, UDK’s language for scripting the game elements and logic. Through scripting we created the many elements and behaviors that comprise the system, both agent model and world objects.

The classes provided by Unrealscript for us to extend have many advantages, but are yet quite complex, with some not-so-obvious connections between components not visible in the last tier of inheritance. As we chose to create a FPS game, we could take advantage of many components (for example use the existing path finding and skeletal animation), and extend the existing classes for Controllers and Pawns to best fit our needs. We also used the bare-minimum base class Object to implement our Behavior Tree and connect it to the extended Controller.

The classes we implement via Unrealscript and mark as “placeable” connect to Editor, where they can be visually placed anywhere in the world, and added to the engine’s update cycle.

The base features provided were helpful yet not complete, therefore some work had to be made to provide extra visual cues such as the item tooltips, bot emotes and HUD, as well as of course the objects needed for our scenario, to best suit our testbed environment.

### 4.2 System Components

Our system required the implementation of different types of components: behavior implementation and setup, and scenario challenges objects. Both of these components consist of scripting and diagramming in Kismet. For the Behavior Tree part of the agent model, we created the individual task objects, which
would bind to the owner controller. We implemented Kismet sequence objects to allow to initialize new
tasks and setup the Behavior Tree structure. We made it so the Bot setup would similarly be done
via Kismet, completely separating configuration from implementation. We created usable items and
challenge objects, connecting the scripting implementation to actions defined via Kismet. While some
challenges consisted on platforms associated with usable objects, others were sentry enemies that
would just pick a target and start shooting, reloading when necessary.

The system components are described in greater detail below.

4.2.1 Bot Setup

The bot setup was mostly done through Kismet, as referenced above in Figure 4.1. We created a Kismet
sequence where at the level start it would bind to a marker object we placed in the map via the Editor,
and spawn a new Pawn, giving it a Controller (brain) and set the assertiveness and cooperativeness
values defined in the box parameterization.

4.2.2 Controller and Behavior Tree

The agent model consists in a Controller that communicates with a Behavior Tree module as was de-
scribed in the Methodology section, as illustrated in Figure 3.2. This Controller holds inside the black-
board of values pertaining his perception of the World Information and the Personality values setup
at its initialization via the Kismet sequenced previously described. There is a two-way communication
between the Behavior Tree’s Tasks and the Bot Controller, such that the Bot feeds to the tree the infor-
mation it needs to calculate the Tasks viability and Utility, and on the tree feeds to the Bot the resulting
decisions of what actions it should perform.

Each of the Behavior Tree’s tasks has the typical task base structure described by behavior trees literature, as was referenced in the State of the Art section 2.4, with the additions of the utility and reordering method as well as the reference to the owner Controller.

![Figure 4.3: Basic task object structure.](image)

We implemented our condition assertion, that defines if a task is viable to be chosen at the time of evaluation, by making use of threshold functions inherent to the task, and the output values of the Utility function, following the model described in the Methodology section.

In order to keep the logic of the behavior detached from the actual implementation of actions, for example, different types of movement, fighting (which could be translated into shooting weapons, melee combat, etc.), the tasks only take care of implementing the logic of behavior choice. After the choice is made they will simply call the corresponding actions methods in the Controller it is binded to, who implements them as best fits the character in question. This way, different creatures can do the same reasoning process but manifest their behavior in their own way.

This separation between reasoning logic and action implementation allows us to easily swap these components with different implementations, as long as we keep the base API.

We separated the setup of the Behavior Tree and the initialization of its composing tasks from the code implementation, through a new Kismet component we created for this purpose, so that this configuration would be more intuitive to adjust, being dynamic, not hard-coded. While we could have read the configuration of the tree from a configuration file to achieve the same separation, by making use of the Kismet tool we create a framework that is much more accessible and easy for anyone to use and extend.

The task initialization boxes connect to one another and successively attach to the previous task, creating the hierarchy of the tree. They also must connect to the controller reference object in Kismet to complete their setup by binding to their owner. The hierarchy can be reconfigured by swapping the connections between task boxes. It is a very convenient tool to visualize the designed tree.

The Behavior Tree that we created for our Bot is illustrated in the Kismet sequence in Figure 4.4

### 4.2.3 Level Objects

In order to facilitate having a world that both the player and the AI companion could easily and correctly perceive, explore and interact with, it was necessary to create various custom objects to populate the
Figure 4.4: Behavior Tree Kismet sequence.
environment.

**Ammo and Health Kits** – we created custom pickup objects for ammunitions and health kits, that could be picked up (Figure 4.5) and added to a custom inventory to facilitate both the interaction with this kind of objects as well as being able to share and keep track of the information regarding the amount of available goods, and their distribution between team members. These essential goods have limited availability and can be seen as the currency of the scenario. Their management directly relates to the well-being of each character. This way we use the different distributions of these goods to effectively manifest each member’s concern for himself or the other party. Each of the team member’s personal inventory is viewed by the team, so that the distribution of assets is clearly known by both the player and Bot. Being considered a kind of currency, these items were also used as prizes for activating platform puzzles, as described ahead.

![Figure 4.5: Pick-up Items.](image)

**Platform-Activated Puzzles** – in order to create challenges that could only be overcome by some form of cooperation, we created platform objects that when triggered would reveal and make accessible prize goods. We designed these elements in a such way that the one who was keeping the platform active (by holding it pressed down, standing on top of it) was not able to reach the prize, giving the opportunity to manifest the different concerns for one’s interests or the other party’s in the choice of who activates, and who collects.

We created a couple of variants for this puzzle: a platform that when active would cause items to reveal themselves (Figure 4.6), a platform that would trigger the opening of a secret passage (Figure 4.7), and the Dual-Activation platform puzzle.

**Dual-Activation Puzzle** – this special type of platform-activated mechanism, requires that both the player and his AI companion stand on top of the puzzle platforms, simultaneously, in order to unlock the
Figure 4.6: Platform puzzle revealing bonus item.

Figure 4.7: Platform puzzle revealing hidden section of the room.
puzzle. The purpose for this puzzle was to add a mandatory cooperative challenge that would benefit not one element of the team in particular, but the team as a whole instead. In our test-bed this was used as one of the final barriers to level completion. Figure 4.8 shows the setup of a Dual-Activation Puzzle.

Figure 4.8: Initial corridor with platform-activated door mechanism and enemies.

*Enemies* – while being animated humanoid elements, these can be considered scenario objects as well, consisting basically in turrets. The enemies pick a target and shoot until they run out of ammo, at which point they reload and go back to shooting their target. These elements served as a challenge to the physical well-being of the team members, hurting them with bullets, which allows for the manifestation of the concern for one’s own skin or the other team-member through the choices on how to handle a dangerous situation: protecting the other by intervening, or hiding when they feel no interest in endangering themselves.

*Level Markers* – in order to facilitate for the AI to perceive the world around it, some extra information was made available to it by adding some special marker objects created to mark recognizable spots in the level (see Figure 4.9). Hiding Spots markers were added to facilitate searching for the closest safe places to try to hide. Exploration Nodes markers were added to help guide the Bot through the level so that it would attempt to explore every node and manage which rooms were already explored.

*Keys and Teleporter* – as a final challenge to the level, we created an exit mechanism that needed to be activated with a set number of Keys. *We created these Key objects in the same way of the Ammo and Health Kits, and the Teleporter trigger would check the character's inventory for the correct amount of these items. The final room where the Teleporter was placed is shown in Figure 4.10.*
Figure 4.9: Markers for Hiding Spots and Exploration Node, and Ammo Kit prefabs in room B.

Figure 4.10: Final room.
4.3 User Interaction

The Bot interacts with the user through its actions in the world, and by expressing in a chat balloon a conversational report of the behavior it is pursuing, which can include directly requesting actions from the player. Additionally, the information regarding the state of the Bot is also conveyed in a box in the Heads Up Display (HUD) interface, allowing the player to keep tabs on the team-mate’s actions even if they get separated or lose line of sight.

The player can additionally interact with the Bot by making explicit requests, such as requesting that the Bot move to the player’s position or requesting supplies. We implemented a set of actions that the player can do to interact with the Bot: “Heal Target”, “Give Ammo”, “Ask for Health Kits”, “Ask for Ammo Kits” and “Call Partner”. While the first two provide benefits to the companion Bot, the remaining consist in requests that favor the player. These last orders are sent to the Bot who is notified of the pending request. The Bot then chooses if they fulfill or reject this order according to their personality and context.

The keys for these actions are shown in the right side of the HUD, and were mapped with usability in mind, placing related actions in the same row of keys.

4.4 Level Design

The design of the scenario level we prepared for the user experience was focused on trying to express the behavior choices at the key situations described by the scenario choices of action for each type of personality in the Experience Design section 3.1.
We present the scenario layout in Figure 4.12.

Figure 4.12: Layout of the scenario level. Green squares represent puzzle platforms. Yellow circles represent the keys the player must gather. Obstacles are presented as blue elements.

Regarding the manifestation of the choice of action when it comes to the distribution of goods between player and Bot companion, we attempted to include key situations where the main focus is obtaining resources. We started our scenario with an event that increased the need for Health Kits, taking the form of an explosion that damages both the player and the Bot. In the initial room (A) we are presented with a limited amount of Health Kits, and there is a clear expression of concerns priorities by the amount of Kits that the Bot will collect or leave for the player.

In addition to presenting limited Health Kit goods in the initial room, we also provide there a cooperation challenge: In the room there is a platform-activated puzzle where if one of the team members steps onto the platform and holds it down, an extra resource is available for the other one to grab. This challenge will once more manifest the preferences of the Bot, who will look at his own state and items, as well as the players' and chose to help or be helped to more items.

Leaving the room will lead to a danger situation in a corridor where two enemies will attempt to shoot at the team. In this situation the choices will be evidenced by either hiding or courageously putting oneself on the line trying to take down the enemies.

After the corridor event, the team members are likely left with a lot fewer ammunitions, and as such we next present a room (B) where the team must choose the distribution of the next limited resource: Ammo Kits.

Another type of Platform-Activated Challenge is provided in the next room (C), where the apparently small room can reveal a hidden section behind a fake wall that can be opened via the platform activation.

We add a couple of more enemies along the way to give the opportunity to the bot to demonstrate his choices of hiding or fighting, that can by now be different, depending on the context of surrounding enemies, own ammunition count, own well-being and the player's.
The final room (E) presents as a final challenge a teleport transport that can only be activated by having three keys. This presents an extra conflict situation where only one element of the team can successfully leave the area. This makes it so that throughout the level when faced with the presence of a Key, a competition situation arises and the Bot will express its levels of Assertiveness by choosing to pickup the item.

Concluding Remarks  In this chapter we described how we implemented our architecture in the Unreal Development Kit (UDK) engine. We developed a framework to easily create new Behaviors and setup Behavior Trees in UDK. This framework benefited from UDK’s tool Kismet, to provide a graphical interface for the construction and configuration of Bots’ Controllers’ Behavior Trees, as well as the setup for the Bot’s personality Cooperativeness and Assertiveness values. We described the game, and its elements, that we implemented as a testbed scenario for the cooperation conflicts challenges.

The next chapter focuses on the experimental process for testing the developed system with users, and results analysis.
Chapter 5

Evaluation

Determining if an agent is truly believable lies in the perception of the users. When evaluated as believable, we can consider the agent to be successfully implemented [18]. This is also true for the perception of personality, so the correct identification of an agent's personality by the users should define the success of this work.

In order to assess the adequacy of our model, testing was conducted with users through sessions of experimenting the testing scenario game.

Each player interacted with a companion agent in the game through four successive runs of the scenario, each with a different pre-assigned personality corresponding to a coordinate on the dual-axis concern model. At the end of the testing session (after having run the level with each of the four personalities, so that the player can compare the experiences), the player was asked to fill a survey where he marked in the model graph where he perceived the bot's personality was situated. Satisfiability and success of the proposed model was assessed by accounting the level of matching between agent personality, as perceived by players, and agents personality initially assigned by the model.

5.1 Evaluation Methodology

Our experimental evaluation consisted in each user interacting with the system and experiencing the scenario in first person, interacting with four different personality bots. Each of the four personalities chosen for the test corresponded to each of the edges on the Cooperativeness-Assertiveness \((c, a)\) graph on Figure 2.1: \((0,0)\), \((0,1)\), \((1,0)\) and \((1,1)\).

Initially, each user was instructed on what the experiment consisted: we were evaluating four different personality bots, where each personality was parameterized according to levels of Assertiveness and Cooperativeness, and we wanted to see if after interacting with each bot in a first-person-shooter type game with some cooperative elements and limited resources, distinct personalities could be perceived, and if differences could be identified, then they were correctly transmitted to the player.

In order for the players to be able to pay more attention to the bot, the experience started with a run through the scenario without the Bot, so they could first familiarize themselves with the virtual
environment. With this practice run we can verify if the player has trouble interacting with the 3D game world scenario, and help the player adjust to the experience. Additionally, this meant that on the following runs the player could focus their attention on the Bot’s actions.

Next, the players played through the scenario with each different bot (different personality values, although using the same model and animations), in a randomized and anonymous order, to prevent any bias related to the ordering of the runs that could impact the players’ impressions.

Throughout running the scenario with each different bot, each player would reflect on what they perceived as the bot’s personality type. After playing through the scenario with each of the 4 bots, each player would finalize their evaluation of each bot’s personality, marking down on the Cooperativeness-Assertiveness graph where they perceived each bot’s personality to correspond to, using as marker the number of the order of the run that bot was played in. The meaning of these concepts of Cooperativeness and Assertiveness, were clearly transmitted to the users as representing concern for the other party’s goals and concern for own goals, respectively, in order to ensure the users’ awareness of these concepts beforehand.

Additionally, we asked the players to describe each of the bot they interacted with in one or two words. We also asked that the players would elect their favorite bot.

Finally the players supplied their demographic information: gender, age group, and experience with games, FPS in particular.

The questionnaire created and used for this purpose can be seen in the Appendix section A.

To achieve statistical significance in our results, we aimed to conduct tests with 30 different players. We were able to perform and gather results from 22 different users.

5.2 User Characterization

Overall the test population consisted of 22 individuals, with ages between 18 and 54 years, mainly male, as shown in the pie charts depicted in Figure 5.1 and Figure 5.2.

![Figure 5.1: Demographic information on the testers age.](image)

These people were volunteers, mostly with a relatively high game experience (Figure 5.3), including the First Person Shooters genre which we chose to implement our game (see Figure 5.4), as well as persistent gaming habits, as characterized by the Figure 5.5. Since the user base was highly familiarized
with the type of game scenario of our tests, the experience was not hindered by difficulties in interacting with the virtual environment.

![Sex](image)

**Figure 5.2: Demographic information on the testers gender.**

![Experience with video games](image)

**Figure 5.3: Demographic information on the testers games experience.**

![Experience with First Person Shooters](image)

**Figure 5.4: Demographic information on the testers experience with FPS games.**
Figure 5.5: Demographic information on the testers gaming habits.
5.3 Experimental Results and Analysis

As mentioned in the previous section, we obtained from the experimental evaluation, data regarding the placement in the Cooperativeness-Assertiveness graph for each tested personality, verbal classification of each personality, player preference and player demographics. In the subsections ahead we describe in detail the results and what could be extracted from the analysis of each of these elements.

5.3.1 Personality Mapping

Figure 5.6 presents the scatter plot containing the raw data results for the users classification of the bots personalities.

![Scatter plot](image)

Figure 5.6: Scatter plot with the results for the user mapping of each of the Bot personalities.

To better view each personality evaluation, we created individualized heatmaps, by plotting the result of applying a Gaussian mixture composition for each personality, as shown in Figure 5.7.

It is apparent from the analysis of these heatmaps that overall the users can differentiate the four different personalities, and correctly evaluate the personalities for (0,1), (1,0) and (0,0) to the right quadrants. The (1,1) personality although well separated from the others in these heatmaps, was placed slightly off its quadrant. It is positive to note, however, that the players correctly perceived that that personality corresponded to an equal balance of Cooperativeness and Assertiveness, as can be seen by the diagonal orientation of its distribution. This displacement may have been caused by some effect where the combination of showing both actions that express high Assertiveness as well as actions that express high Cooperativeness, slightly counter the perception of each of the individual axis values.
Figure 5.7: Heat map of the results for the users mapping of the perceived Bot personalities.
Quadrant Matching

One approach to evaluating the validity of our solution is analyzing the results to check if the users correctly evaluated the bots personalities to their right quadrant. Ideally, each of the (0,1), (1,1), (1,0) and (0,0) personalities should be matched to the corresponding quadrants.

This matching can be represented as a confusion matrix, showing for each of the assigned personalities the distribution of the users evaluation between the different quadrants. Ideally this matrix should be diagonal (with value 1), presenting no incorrect assessments. Non-zero values outside the diagonal reveal a mismatch between the assigned and perceived bot personality quadrant.

Experimental results are shown in Figure 5.8.

From the confusion matrix, $C$, depicted in Figure 5.8 we can see a trend where while personalities with the same value of Cooperativeness are sometimes confused between one another, there is a clear distinction between different Cooperativeness valued personalities. Different values of Assertiveness, while differentiated, seem not to be as clearly perceived (see light orange-highlighted values). Taking into account that the distribution of a random evaluation of all the personalities should yield a 0.25 assignment to each of the quadrants, our results can be viewed as positive, since we have correct assignment percentage consistently above $\sim 60\%$ (see green-highlighted values). From this confusion matrix we can estimate that the overall error probability $\left(1 - \sum_{i=1}^{n} C_{ii}/n\right)$ is 32.9%.

Looking into differentiating the Assertiveness values (see Figure 5.10), the error probability is 23.9%,
higher than for the Cooperativeness differentiation, although lower than the error for the four class discrimination, as expected.

**Sample Distribution Normality**

In order to verify that the data for each personality follows a Normal Distribution, so that we can apply statistical tests that assume this underlying hypothesis, we used the Kolmogorov–Smirnov test provided by Matlab’s Statistics Toolbox. The results of this test are presented in Table 5.1 and Table 5.2, for the Cooperativeness and Assertiveness dimensions, respectively.

<table>
<thead>
<tr>
<th>p-value</th>
<th>Cooperativeness</th>
<th>Assertiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 0)</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td>(0, 1)</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>(0, 0)</td>
<td>0.24</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: p-value results for the Kolmogorov–Smirnov test for the Cooperativeness dimension.

<table>
<thead>
<tr>
<th>p-value</th>
<th>Cooperativeness</th>
<th>Assertiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 0)</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>(0, 1)</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>(0, 0)</td>
<td>0.12</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: p-value results for the Kolmogorov–Smirnov test for the Assertiveness dimension.

We can see that for each of the p-values they are above the 5% significance value, so we should not reject the null hypothesis of Distribution Normality for each of the independent data for Cooperativeness and Assertiveness.

**Two-Sample t-Test**

We further evaluate this differentiation between types of personality by applying a two-sample t-test, with a p-value of 5% significance, comparing the different perceived personalities data, where we test the null hypothesis that the data in the two personalities x and y comes from independent random samples from normal distributions with equal means and equal but unknown variances. Rejecting the null hypothesis means that the two personalities come from populations with unequal means. We applied this test by using Matlab inbuilt function *ttest2*, comparing the Cooperativeness and Assertiveness dimensions separately.

In Table 5.3 we present the p-values resulting from the ttest2 for the Cooperative dimension, and in Table 5.4 for the Assertiveness dimension. If the p-value is above the significance level (5%) then we do not reject the hypothesis, otherwise we do.

The results presented in the Table 5.3 show that by using the Cooperativeness values alone the players can differentiate the behaviours with high Cooperativeness values ((1, 0) and (1, 1)) from the behaviours with low Cooperativeness ((0, 0) and (0, 1)).
Table 5.3: p-values Results for the t-test for the Cooperativeness dimension (rows represent the assigned personality, columns represent the perceived personality).

<table>
<thead>
<tr>
<th></th>
<th>(1,0)</th>
<th>(1,1)</th>
<th>(0,1)</th>
<th>(0,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1.000</td>
<td>0.564</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>(1,1)</td>
<td>0.564</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>(0,1)</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.421</td>
</tr>
<tr>
<td>(0,0)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.421</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 5.4: p-values Results for the t-test for the Assertiveness dimension (rows represent the assigned personality, columns represent the perceived personality).

<table>
<thead>
<tr>
<th></th>
<th>(1,0)</th>
<th>(1,1)</th>
<th>(0,1)</th>
<th>(0,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.798</td>
</tr>
<tr>
<td>(1,1)</td>
<td>0.000</td>
<td>1.000</td>
<td>0.356</td>
<td>0.008</td>
</tr>
<tr>
<td>(0,1)</td>
<td>0.000</td>
<td>0.356</td>
<td>1.000</td>
<td>0.001</td>
</tr>
<tr>
<td>(0,0)</td>
<td>0.798</td>
<td>0.008</td>
<td>0.001</td>
<td>1.000</td>
</tr>
</tbody>
</table>

We can see in the results pertaining the Assertiveness statistical analysis (Table 5.4), that by using the Assertiveness values alone the players can discriminate the high Assertiveness behaviours ((0, 1) and (1, 1)) from the low Assertiveness ones ((0, 0) and (1, 0)).

These results further validate that there is a clear and statistically significant separation of personalities along the two axis.

**Personalities Distributions**

We now focus in the users ability to consistently classify the four different personalities, regardless of the alignment between perceived and assigned personalities. In order to do this, we address this as a statistical classification problem, where each of the four personalities corresponds to a class. We will be applying a MAP (Maximum a posteriori) decision rule, using a leave-one-out error probability estimation. Two classifiers were implemented for this purpose by exploring two different ways of estimating the class conditional probability density functions: (a) fitting a bivariate Gaussian distribution; (b) using a Parzen Windows estimator. See Appendix B for more information on the process of applying this statistical analysis. We start by using a Gaussian distribution as base for the classifier model since it is the most simple and commonly used distribution, that generally proves to be robust for most data. We then attempt to improve the analysis results by applying a non-parametric approach, namely the Parzen Windows estimator.

**Gaussian Distribution Approximation.**

Regarding the Gaussian Distribution classifier, we obtained the class regions shown in Figure 5.11. The resulting confusion matrix, shown in Figure 5.12, leads us to conclude that the Gaussian distribution assumption correctly approximates the class conditional probability function was not very accurate, as the results are worse than for the simple quadrant discrimination, with an overall probability error of 37.5%.
Parzen Windows Approximation.

The mapping of decision regions obtained from the classifiers for the Parzen Window estimator is shown in Figure 5.13.

The resulting confusion matrix presented in Figure 5.14, yields more positive results, with an error of 31.8%.

While this result improves over the raw data confusion matrix, the improvement is not very significant, which demonstrates that there is room for improvement both in terms of accurately conveying the personalities correctly in absolute terms of the quadrants, but also in a way that is perceived consistently and distinctively by the users. This should be part of Future Work by tuning the model parameters and behaviors, described in Chapter 3.
Figure 5.13: Classifier regions for the Parzen Windows distribution approach.

Figure 5.14: Confusion matrix for the Parzen Windows classifier approach.
5.3.2 Personality Description

We asked the users to describe in one to two words each of the bots they played with. There were a few outliers to this subjective evaluation, but the general verbal description seems to be consistent with what was intended to convey to the players. For example, for the Accomodative bot (1,0), the general description was of an altruistic companion, as pointed out by terms such as: “considerate”, “mindful”, “reckless”. For the opposite personality (0,1) for the Competitive bot, adjectives include: “individualist”, “uncooperative”, “jackass”. The Collaborative Personality (1,1) was perceived as: “helpful”, “balanced”, “fair”. Finally, the Avoiding personality (0,0) was assessed as: “stupid”, “zero”, “lazy”. The complete list of verbal evaluation for each of the personalities can be found in Appendix C.

These characterizations of the bots reflect a positive impression regarding the believability of the bots, since the human adjectives chosen to describe the personalities show some empathic and emotional connection with the bots.

5.3.3 Players’ Bot Preferences

As a flavor factor, we present here in Figure 5.15 the player preferences regarding the different personalities demonstrated in the experiment.

![Histogram of players' favourite bot personalities.](image)

It seems natural that most players like the most balanced personality the best, since it seems to exhibit the most fair behavior choices.

It is interesting to see, however, that unlike what we expected, the players next preference is not for a more Cooperative bot, but instead the most Assertive. This is possibly due to the fact that the more Assertive Bot appears to display more initiative by collecting all the items it finds and moving on its own, instead of leaving everything for the player to collect, and players like this more active type of teammate. This preference is certainly also affected by the fact that the most Cooperative Bot often ends up dead
due to the fact that the more Cooperative bot puts the player's needs above its own (the bot does not actually care for itself), and players see the bot as a hindrance to the team. It is no surprise that the least preferred personality was the Avoiding one, since it is the bot with less initiative and presents itself as useless to the players, having actually instigated the feeling of frustration on them.

**Concluding Remarks**  From the analysis presented in this chapter, we can see that our results seem to verify that the players successfully differentiate different personalities. In particular, when looking into each of the the Cooperativeness and Assertiveness dimensions separately, the players clearly distinguish the high values of the dimension from low values. There is room for improvement when it comes to accurately identifying the correct combination of Cooperativeness and Assertiveness of the assigned personalities. This issue could be addressed by further tuning the model parameters. The fact that the players were able to relate to the Bot as a living team mate throughout the game experience, and describe them verbally with human characteristics and adjectives (actually describing what they felt the bot was thinking as if it was a real entity with its own interests during playtesting), seems to assert the believability of the developed character.
Chapter 6

Conclusions and Future Work

There are many ways to design deep and engaging characters for virtual games, and there is still need for improving the existing solutions as the game industry grows more demanding for believability. In this work we explored a dual-concern (concern for self – Cooperativeness – and concern for the other – Assertiveness) conflict-handling model to produce believable and distinguishable bot personalities in a companion cooperation experience set in a First Person Shooter (FPS) game. We conceived a model based on Behavior Trees combined with Utility evaluation, turning the traditionally static structure of Behavior Trees into a more dynamic system, by reordering the structure at key moments of evaluation. We implemented our model in the Unreal Development Kit engine, applying it to a scenario testbed FPS game that we created in the same platform, where we had to add elements specifically created from scratch to adequately fit our problem of cooperative interactions. When creating the model and testbed, we ended up developing a tool for UDK for creating and setting up Bots based on a Behavior Tree model. Special focus was placed on designing a highly parameterizable model and implementation, allowing for fine-tuning and incremental improvements to the fitting of the personalities expression.

We tested the solution with users, in sessions of playthroughs with various bot personality setups. Experimental results have shown that the proposed solution successfully expresses distinct personalities, where the players can clearly distinguish high Cooperativeness from low Cooperativeness, as well as high Assertiveness from low Assertiveness. However, there is still room for improvement, as the matching between the defined personalities and perceived personalities could be increased.

Given the parameterizable nature of the proposed and implemented solution, future work could include an automatic procedure for incrementally calibrating the parameters to best express the assigned personalities. This automation would require that many users run the experiment level to fine tune these values, so it would need a very large player base and ideally simultaneous runs in order to not be a very slow process.

Another research direction for future work can be the adaptation of the values defining the bots personalities to best meet the users preferences. This would involve inferring the players personalities and adapting the companion AI to match what kind of personality is generally preferred by that type of player.
A different approach to expanding our research could be the extension of our model to include in the team model, not only player and bot, but also any number of other team-members, including other bots. This could lead to very interesting interactions between a more complex team, and possibly allow for greater differentiation between the different team-members personalities expression.
Bibliography


Appendix A

Survey Form

CoBelievable
This survey will allow to evaluate the validity for the Bot’s model that relates personality with its behavior.

* Required

Demographics

Age *

Sex *

Experience with video games *

1 2 3 4 5

Very Little ○ ○ ○ ○ ○ A Lot

Experience with First Person Shooters *

1 2 3 4 5

Very Little ○ ○ ○ ○ ○ A Lot

How many hours weekly do you spend playing games? *

Figure A.1: First part of the web survey.
CoBelievable -- Survey (Part II)

Please mark in the graph where you felt that each Bot’s personality was set (use the number of the order of the bot as the mark placement symbol)

Please describe each bot in one or two words:

1.

2.

3.

4.

Figure A.2: Printed part of the survey.
Figure A.3: Second part of the web survey.
Appendix B

Bayesian Classifiers

Let \( \Omega = \{ \omega_1, \omega_2, \omega_3, \omega_4 \} \) be the set of classes representing, respectively, the set of personalities (0,1), (1,1), (1,0) and (0,0).

From each class of perceived personality results, as shown in Figure 5.7, we fit a class conditional probability density function \( p(x|\omega_i) \), representing the likelihood of observing \( x \) in class \( \omega_i \), where \( x \) is a two dimensional vector with coordinates (cooperativeness, assertiveness).

Two estimators are issued:

1. **Parametric Estimator** – In its simplest form, we assume that each class can be described by a bivariate gaussian distribution, \( p(x|\omega_i) = N(\mu_i, \Sigma_i) \), with \( \mu_i \) the vector of mean values, and

   \[
   \Sigma_i = \begin{bmatrix}
   \sigma_{cc} & \sigma_{ca} \\
   \sigma_{ac} & \sigma_{aa}
   \end{bmatrix},
   \]

   the covariance matrix.

2. **Parzen Estimator** – In this case we assume no particular distribution for the data, but estimate the distribution using a Parzen windows estimator, with gaussian kernel, with each observation \( x_i \) contributing for the overall probability density with a gaussian of the form \( N(x_i, 0.075I) \), with \( I \) being the identity matrix, leading to the fitted class conditional probability density:

   \[
   p(x|\omega_i) = \sum_{i=1}^{n} N(x_i, 0.075I)
   \]

Given the estimated class conditional probability densities, \( p(x|\omega_i) \), and the a priori class probabilities, \( P(\omega_i) \), the MAP classifier assigns an arbitrary observation \( x \) to the class of maximum a posteriori probability, namely:

\[
\omega_i : i = \arg \max_j P(\omega_j|x),
\]

which, according to the Bayes rule, is given by

\[
P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}.
\]
Since in our experiments all the personalities (classes) have equal a priori probability, the MAP classifier is equivalent to the Maximum Likelihood classifier, with decision rule:

\[ x \in \omega_i : i = \arg \max_j P(x|\omega_j). \]

For the above classifier, and for each of the probability density estimator methods, we estimate the corresponding confusion matrix and overall error probability, using the leave-one-out method: when classifying each sample, we remove the sample from the computation of the class's conditional distribution, in order to avoid biasing the error estimation to a more favorable result.

This classifier, as well as the probability density estimators, were implemented in Matlab and applied to our experimental data.
Appendix C

Verbal Classification of the Bots’ Personalities

<table>
<thead>
<tr>
<th>(1, 0)</th>
<th>(1, 1)</th>
<th>(0, 1)</th>
<th>(0, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>babysitter</td>
<td>ideal companion</td>
<td>individualist</td>
<td>uncooperative asshole</td>
</tr>
<tr>
<td>considerate</td>
<td>incompetent</td>
<td>doesnt care</td>
<td>all his</td>
</tr>
<tr>
<td>retarded</td>
<td>friendly</td>
<td>uncooperative</td>
<td>stupid</td>
</tr>
<tr>
<td>brain dead</td>
<td>helpful</td>
<td>eager</td>
<td>dick</td>
</tr>
<tr>
<td>careless</td>
<td>helpful</td>
<td>brave</td>
<td>selfish</td>
</tr>
<tr>
<td>noob</td>
<td>balanced</td>
<td>helpful</td>
<td>uncooperative assertive</td>
</tr>
<tr>
<td>brain dead</td>
<td>the best</td>
<td>helpfull with iniciative</td>
<td>not helpfull at all</td>
</tr>
<tr>
<td>egotist</td>
<td>dependent</td>
<td>normative</td>
<td>psychopat</td>
</tr>
<tr>
<td>selfish</td>
<td>too cooperative</td>
<td>average</td>
<td>weird choices</td>
</tr>
<tr>
<td>oblivious of surroundings</td>
<td>most effective</td>
<td>hoarder</td>
<td>stubborn</td>
</tr>
<tr>
<td>helpfully reckless</td>
<td>balanced</td>
<td>Self absorbed</td>
<td>Dexters sister</td>
</tr>
<tr>
<td>mindful reckless</td>
<td>fair</td>
<td>distracted aloof</td>
<td>selfish</td>
</tr>
<tr>
<td>altruistic</td>
<td>douchebag</td>
<td>hero</td>
<td>hero</td>
</tr>
<tr>
<td>balanced</td>
<td>independent</td>
<td>stubborn</td>
<td>helpful</td>
</tr>
<tr>
<td>lively</td>
<td>lonely</td>
<td>goodfella</td>
<td>sad</td>
</tr>
<tr>
<td>OK</td>
<td>cooperative scared</td>
<td>helpfull lazy</td>
<td>zero</td>
</tr>
<tr>
<td>well intentioned</td>
<td>buddy</td>
<td>Jackass</td>
<td>lazy</td>
</tr>
<tr>
<td>over cooperative</td>
<td>balanced</td>
<td>little cooperative</td>
<td>useless</td>
</tr>
</tbody>
</table>

Table C.1: Verbal evaluation of the bot personalities.