CoBelievable – The Effect of Cooperation in Believability

Ana Patrícia Fred Filipe
Instituto Superior Técnico
15th October 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

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.

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].

Personality, being a very important aspect of designing believable game agents, has a great impact in social interactions, where it can be noticed most. 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.

1.2 Goals and Work Outline

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.

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 analysed, in order to verify
the quality of the model and validate the proposed solution.

2 State of the Art

2.1 Conflict Theory

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.

Researchers have evolved different conflict handling models, some of the most relevant:

- Deutsch’s Social Interdependence Theory [2], explores the types of interdependence of goals (positive or negative) between the party, comparing effects of cooperation versus competition in conflict.

- The Social Motivation Theory [3] compares the different social motives of the involved parties, focusing primarily in pro-self versus pro-social motives.

- 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 defines five handling modes: competing, collaborating, compromising, avoiding, and accommodating. [4]

2.1.1 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. [5]. 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.

Another example of a project that relates to our problem is “My Dream Theatre” [6], a serious game developed with the purpose of teaching conflict resolution to children. 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 [7], 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.

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 [8].

Personality Personality relates to the motivations of behavior and regulates how and which goals are pursued. In order to be believable, an agent is not required to choose the best plans and actions, but it is imperative that its every action shows its own style and personality [7].

There are different approaches to define and characterize personality types, one of the most
relevant being the Myers-Briggs model.

This model describes four dichotomies: Sensing-Intuition, Thinking-Feeling, Judging-Perceiving, and Introversion-Extraversion. [9]

Thomas and Kilmann [10] have studied how the Jungian personalities of the Myers-Briggs model relate to the different conflict management modes. In the study a model is proposed mapping the introversion-extraversion and feeling-thinking dimensions to the integrative dimension and distributive dimension of conflict behavior (described in the Conflict-Handling Modes Section) respectively.

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, portraying characters in the selfish to altruistic spectrum.

2.3 AI Models

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

Finite-State Machines (FSM), very commonly used for behavior modelling, break down a character’s behavior into states. 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 [13].

Goal-oriented planners, provide an architecture based on collections of hierarchical goals [13], which can consist on a single action, or be a composite, describing more complex tasks. Actions are associated with preconditions and effects, and each desired goal is pursued by working out which actions must be completed to achieve it. 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. [14].

Behavior trees have become a popular structure for creating AI characters [15], being composed of Tasks which can be: Conditions, Actions and Composites. Conditions establish tests that must be checked in order to successfully execute an Action. Composites make up the branches, usually either Selectors or Sequences, where Selectors encompass alternative actions, and 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.

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 [16]. 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.

3 Approach and Architecture

We aimed to create an interactive experience where the player interacts with the bot in first person in order 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.

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

3.1 Experience Design

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.

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.
Having chosen to design a companion bot, we should create experiences where the bot and player buddy-up to progress through the level.

The following behaviors are expected from each style of conflict management:

*Avoiding* – sometimes performs only easy, low-consequential tasks; will not care for the player’s concerns, nor its own; won’t make efforts to help the team’s objectives

*Competition* – agent will disregard player’s intentions; will follow its own agenda; will keep advantages for itself; 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; both agent’s and player’s objectives are equally important; will divide resources with fairness

### 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 2.

![Figure 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.

**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.
Utility Evaluation Each task has an Utility function that depends on the world state and personality (the position in the conflict model’s two dimensions, of cooperativeness and assertiveness, which was assigned to the character on its creation). 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 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.

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.

\[ U_{\text{action}}(c, a) = f(c, a, F) \]  \hspace{1cm} (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.

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.

Reordering of the Tree 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.

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 to build our Agent Model upon.

The testbed scenario also benefitted 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.

4.1 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.

Controller and Behavior Tree The agent model consists in a Controller that communicates with a Behavior Tree module as was described in the Methodology section, as illustrated in Figure 2. 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 information 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, with the additions of the
utility and reordering method as well as the reference to the owner Controller.

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.

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.

The task initialization boxes connect to one another and successively attach to the previous task, creating the hierarchy of the tree. The hierarchy can be reconfigured by swapping the connections between task boxes. It is a very convenient tool to visualize the designed tree.

**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 environment.

*Ammo and Health Kits* – we created custom pickup objects for ammunitions and health kits, that could be picked up and added to a custom inventory to facilitate interacting with these objects and keeping track of their distribution.

*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 (Figure 3).

Figure 3: Plaftorm puzzle revealing hidden section of the room.

*Enemies* – these objects consist basically in turrets, presenting a danger which allows for the manifestation of the concern for one’s own skin or the other team-member.

*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).

Figure 4: Markers for Hiding Spots and Exploration Node, and Ammo Kit prefabs in room B.

### 4.2 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. Additionally, the information regarding the state of the Bot is also conveyed in a box in the Heads Up Display (HUD) interface.

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”. The Bot then chooses if they fulfill or reject this order according to their personality and context.

### 4.3 Level Design

We present the scenario layout in Figure 5.

Figure 5: Layout of the scenario level.
In the initial room (A) we start the scenario with an explosion that damages both the player and the Bot and we limited available Health Kits, forcing expression of concerns through resource distribution.

We also provide there a cooperation challenge: In the room there is a platform-activated puzzle where an extra resource is available for the one of the team mates to grab.

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 left with a lot fewer ammunitions, and 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.

5 Evaluation

Determining if an agent is truly believable lies in the perception of the users. 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 twenty-two 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

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 distinct personalities could be perceived, and if differences could be identified.

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.

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.

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.

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

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

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, volunteers, mostly with a relatively high game experience, including the First Person Shooters genre which we chose to implement our game. Since the user base was highly familiarised with the type of game scenario of our tests, the experience was not hindered by difficulties in interacting with the virtual environment.

5.3 Experimental Analysis

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 6.

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.

**Quadrant Matching** One approach to evaluating the validity of our solution is analysing 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 7.

From the confusion matrix, $C$, depicted in Figure 7 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. 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 ~60%.

Looking into differentiating the Assertion values, 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 we used the Kolmogorov–Smirnov test provided by Matlab, where the resulting p-values 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 differentiation between types of personality by applying a two-sample t-test, with 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. We applied this test by using Matlab.
inbuilt function \texttt{ttest2}, comparing the Cooperativeness and Assertiveness dimensions separately. In Table 1 we present the resulting p-values. If the p-value is above the significance level (5%) then we do not reject the hypothesis, otherwise we do.

\begin{tabular}{cccc}
(1, 0) & 1.000 & 0.564 & 0.000 \\
(1, 1) & 0.564 & 1.000 & 0.000 \\
(0, 1) & 0.000 & 0.000 & 1.000 \\
(0, 0) & 0.000 & 0.421 & 1.000 \\
\end{tabular}

Table 1: p-values for the t-test for the Cooperativeness dimension.

The results presented in the Table 1 show that by using the Cooperativeness values alone the players can differentiate the behaviours with high Cooperativeness values from low Cooperativeness.

\begin{tabular}{cccc}
(1, 0) & 1.000 & 0.000 & 0.000 \\
(1, 1) & 0.000 & 1.000 & 0.356 \\
(0, 1) & 0.000 & 0.356 & 1.000 \\
(0, 0) & 0.798 & 0.008 & 1.000 \\
\end{tabular}

Table 2: p-values for the t-test for the Assertiveness dimension.

We can see in the results pertaining the Assertiveness statistical analysis (Table 2), that by using the Assertiveness values alone the players can discriminate the high Assertiveness behaviours from the low.

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 will 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. We started 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. With this approach we obtained clear separate classified areas for the four personalities, however the values associated with the resulting confusion matrix led us to believe 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%.

We then attempted to improve the analysis results by applying a non-parametric approach, namely the Parzen Windows estimator.

The resulting confusion matrix presented in Figure 8, yields more positive results, with an error of 31.8%.

![Figure 8: Confusion matrix for the Parzen Windows classifier approach.](image)

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.

5.3.1 Players’ Bot Preferences

As seems natural, most players like the most balanced personality the best. It is interesting to note, however, that unlike what we expected, the players next preference is not for a more Cooperative bot, but instead the most Assertive, possibly due to the fact that the more Assertive Bot appears to display more initiative by collecting all the items. It is no surprise that the least preferred personality was the Avoiding one, since it will be barely active and avoid interacting.

6 Conclusions

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.

References


