AI Behaviour Controller in Unity

Yanzhou Yang

Department of Computer Science

University of Western Ontario

December 05, 2021

Supervisor: Professor Mike Katchabaw

1. Introduction

Artificial intelligence in video games has been a broad area of study since the 1970s, and the main idea of the study is to make the game more entertaining(Rabin, 2002), and make the player feel more immersive in the game world. Among all of the different researches of AI in video games, how to make bots and characters in game behave more believable is a more active and important area of study. How the non-playable characters(NPCs) would react to different events and stimulus in the game world is an important factor of players' experiences of the game, it is especially true for games that expect players to have countless interacts with those NPCs during the game process such as story-driven genre games, or open world games(Bailey, You, Acton, Rankin & Katchabaw, 2012), because whether the players can feel immersive and enjoy the game is largely depends on if NPCs can make believable reactions to events or stimuli in-game.

Making the NPCs' reaction believable or creating believable bots/characters, however, is an incredibly tough problem, because the process of creating believable reactions to stimuli or events under different situations would be pretty comprehensive since the developers should consider not only the current status of the game character but also the status of the game world. For example, when the player's "relationship" with a character changes, that character should not have the same reactions to the player as before, and characters should not react identically to the same stimuli in the game after certain game events have happened . That is saying to make characters believable, a character should have its unique "personality" and "relationship" with other characters and players in the game. So creating a believable character is beyond the area of computer science, it also includes the use of psychology and sociology models.

And the AI Behaviour controller in Unity is an Unity asset that can help the game developers to create and manage believable characters in game, which includes editing their “personalities”, current status, and their reactions to certain events and stimuli in the game world.

2. Related works

Character's behavior in video games is often determined by using finite state machines or decision trees, which will make characters loss believability, it has been argued that enhancing the personality, emotions, moods and the social relationship of the character can make the characters behavior more diverse and hence more believable(Kersjes & Spronck, 2016).

Kersjes and Spronck's work has shown the use of Psychology models(Personalities, moods, and emotions)can increase the character's believability, while another commonly used way to make the characters more believable in video games is the use of Reputation systems. Reputation system refers to the system that manages the opinion of certain groups of characters to the player, and you can see reputation systems in almost all Role-playing or Open world games, such as "Assassin's Creed","Grand Theft Auto" and " The Elder's Scroll". The reputation system uses the sociology models to divide characters in game into different "roles" such as guards, civilians, or merchants, and according to the player's action, the reputation system would change the opinion of characters with certain "roles" to the player immediately(Brown, Lee & Kraev, 2021). For example, as the screenshot taken from "Grand Theft Auto V" in Figure.1, when the reputation system detects that the player attacked other NPCs, the opinion of all of the "Police officers" in the game to the player would be changed simultaneously, and they will all

come to arrest the player.

Figure.1 Player is attacked by the “police” in GTA:V

The problem of the reputation system is obvious, the behaviour patterns of the characters in the game would solely depend on their “roles”, it makes all of the characters of the same role react identically all over the whole game world, which would destroy the immersion of any player.

Our work, the first generation of AI Behaviour Controller, is a specialized reputation system, it uses the idea that to make characters more believable, characters should react differently to certain stimuli in the game based on their unique personalities (Rosenthal & Congdon, 2012). The system would divide the characters in the game world into different “classes” , and also give each character an unique personality. Whenever the character detects a stimulus, the system would not assign a global reaction to all of the characters from the same

"class". Instead, it would provide each character with a list of possible reactions of that stimuli, and each character would choose the reaction that fit its own personality most from the list. As a result, the AI Event Controller increases the diversity of characters' reactions to the same stimuli and suspends the player's disbelief, and thus make the game more entertaining.

3. AI Behaviour Controller in Unity

3.1 System overview

As shown in Figure.2, the system of AI Behaviour Controller is consist of 5 parts, the "Manager" class would act as the database and the controller of the whole system, it stores each and every piece of data in the system, including the current status of all characters and events in the game world and all of the personality traits in the system, the "Characters' Class" class is a container of different groups of characters and events in the system, each character in the game world would belongs to at least one "Character Class" while each event can be used by several character's classes at the same time. The " AIEvent" and "Character" class stores the current status of each event and characters in the game, and those "events" are the possible reactions to certain stimuli in the game world. And lastly,when a character detects stimuli, the randomized system would calculate the weights of all possible reactions, and choose a reaction for the character.

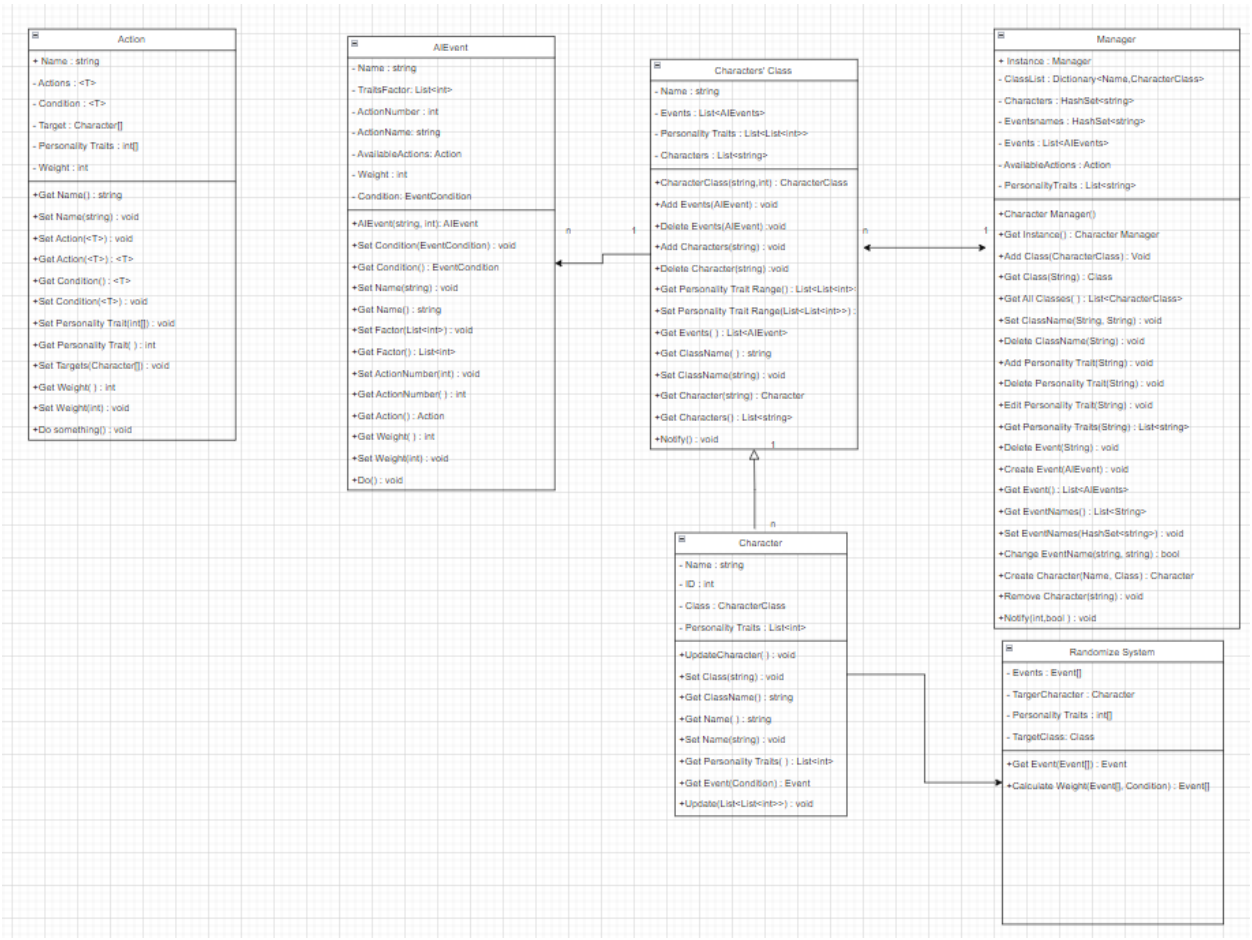

Figure.2 Class diagram of the AI Event Controller

3.2 How it works

Unlike the regular reputation systems, the Ai Behaviour controller would not generate a global reaction to certain stimuli/events to a group of characters. Instead, each character will decide it's own reaction to the stimuli individually, when a character detects a stimulus, it would ask it's character class for a list of possible reactions(events) to that stimuli, according to the different types of stimuli(Event condition), the character class would return a set of possible events to that character, and finally, the randomize system would choose one of those event from

the possible events list based on the “personality” of that character, and return the event, so different characters from the same “character class” would react differently to the same stimuli, and their reactions would be based on their unique “personality”. For example, if there are two characters, one named “Timid Guard” and the other one is “Brave Guard”, they both belong to the “Guard” class, and two events “Fight” and “Run away” are stored in the “Guard” class, so whenever one of those two characters has been attacked, the “Brave Guard” would be more likely to choose the “Fight” event as its reaction, while the “Timid Guard” would choose to run away. And in the AI Event Controller, the “Personality” of each character will be defined by a set of “Personality traits” values, since each game will need different sets of personalities, so the name and the number of those “Personality traits” in the controller can be edited by the game developers to make the controller fits their games.

3.3 Classes review

3.3.1 Manager

As its name suggests, the “Manager” class would work as the system manager and also the database of the AI Behaviour controller, every time if there is a change of the data(adding, deleting and editing) in the system, it needs to go through the “Manager” class. For example, if a game developer wants to add an new event or a new characters to the character class “A”, then the character class “A” would call the “create event” or “create character” method in the Manager class, the Manager would create the event or character object as requested, and then save the new created object to its list and send to character class “A”. By doing that, the developers can track or edit the data of any characters or events stored in the system via the system manager without checking each of the character classes manually. The “Manager” class is

designed using the Singleton design pattern, to make sure there is only one instance of the "Manager" exist at the same time, and it will also provides a global instance of the Manager that can be accessed by all of the other objects in the system, and it also uses the idea of the Observer design pattern, whenever the data stored in the Manager changes, the Manager will call its "Notify" method to inform all of the character classes about the change. For example, if the developer adds a new Personality trait to the system, then the Manager will call the Notify method, and each character class in the system will add this new traits to its trait list.The data stored in the Manager class would be exported to a local XML file to ensure the data persistency.

3.3.2 Character's Class

The Character's class is a container for a group of characters and events in game, each character class will have a unique set of personality trait ranges, for each character in the system, the values of each of its personality traits would be a random value within the ranges defined by its character class. And the Character's class also uses the idea of the Observer design pattern, whenever the data of the character class has been edited, the class will notify all of its characters about the changes. For example, if there is a new personality trait added to the class or one of the traits' range has been changed, the class will send the new set of ranges to all of its characters, then each of the characters will update or generate a new set of personality trait values based on the new set of ranges.

3.3.3 Characters

"Characters" class is the class for all characters in the game, it stores the current status of the character including the name of the character, the ID of the character, the class of the character and the personality traits values of the character. In the AI Event Controller, characters

are allowed to have the identical names, but each character will have a unique numeric ID, the developers can use those IDs to find and edit the specific character they are looking for, and since the properties of the characters will be game specific, so the Character class is an abstract class, that needs the game developers to specialized to fit their needs before use.

3.3.4 Events

The “Events” are the possible reactions of characters to certain stimuli in game, each event has an event name, a set of personality traits, event condition, and actions. The name of each event would be unique in the system, event condition defines the specific types of stimulus that the events could be triggered, and the character class would use those condition to choose a set of possible events whenever the character detects a stimulus , actions are the actual reaction to the stimuli, and the values of personality traits of the event are actually the factors used by the randomized system when calculating the weight of the event. If the value of trait A is higher than other traits of the event, then the weight of this event would largely depend on the value of the character’s personality trait A. Let's recall the example in Section 3.2, if there are two character “Brave Guard” and “Timid Guard” belongs to the character class “Guard”, and there are two events “Fight” and “ Run away” in the “Guard” class as shown in Figure.3, then for the “Brave Guard”, the weight of event “Fight” would be calculated as : 8*5 + 2*1 = 42, and the weight of event “Run away” would be 1*8+5*2 = 18, since the weight of event “Fight” is higher than the weight of event “Run away”, when the “Brave Guard” detects a stimulus, it would be more likely to fight rather than choose to run away, and for the “Timid Guard”, it will choose to run away more often than fight. And as the “Character” class, the “Event” class are also abstract classes since the needs for the structure of the events of various types of games are different, the event

class will also need the game developers to specialize before use.

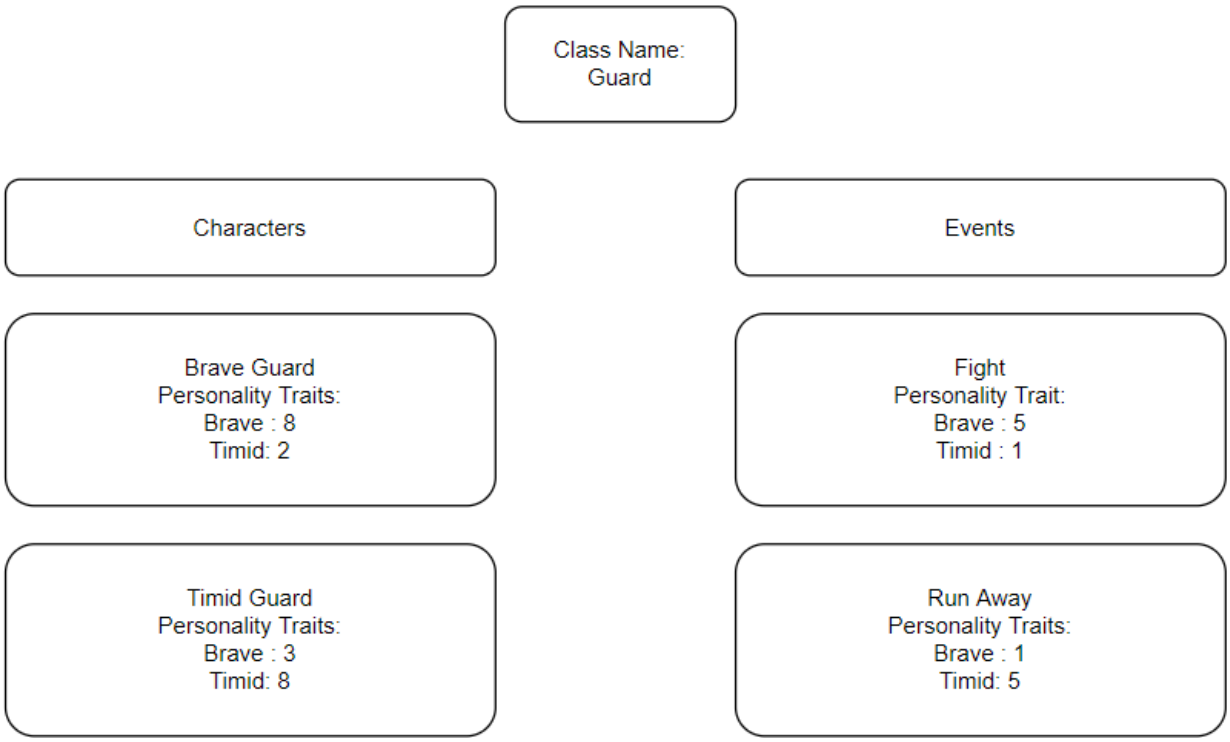

Figure.3 Character class example

3.3.5 Randomized System

We expect that characters would have a believable and dynamic response to every relevant stimulus in game, so when a character detects a stimulus, it would send the stimuli to its class and choose a proper reaction to the stimuli from a set of possible events. The “Randomized System” plays the most important role during this process, after the character send the stimuli to its class, the class would choose a set of possible events of this type of stimuli from its event list, and the randomized system would need to choose a proper event for the character from all of the possible events, it would first receive a package that includes the set of possible events and the current status of the character, and then, according to the values of personality traits of the

character, the randomized system would calculate the weight of each possible events, the formula used to calculate the weight can be customized by the game developers. And after the weight calculation, one event will be selected and returned to the character as the reaction of the character to the stimuli by using a weighted randomized formula. That is saying, after the weight calculation, the events with higher weight will have more possibility to be returned to the character, but the events with lower weight will also have a chance to be selected as the reaction to the stimuli.

4. Challenges and Results

4.1 Challenge

Since the AI Behaviour controller needs to work as an Unity asset, study the Unity Editor API is the absolutely necessary but the most time-consuming effort during the development process, The Unity Editor API has been reworked and updated multiple times in recent years, some of the tutorials and resources online are out-dated, and the official tutorial published by Unity Learn Team does not cover much about the use of the Unity Editor API.

For most games, there are huge differences in the way of design and development, for example, in some games, the ability of the character can be implemented as a method in the script, while the others may implement the ability as a component attached to the character object or even extend the ability as a new script. And those differences make it hard to design a controller that can fit the needs of most of the games at the same time.

4.2 Result

Since there are too many contents need to be game-specific, the AI Behaviour controller ends up being designed as most classes are abstract classes, they would only contain necessary

properties and methods that need to be used by the “Manager” class, for example, the “Character” class contains only 4 properties: name, ID, class and personality traits, which are the essential properties of a “character” that the Manager needs, and the game developers can extend the “Character” class as they want in the future, such as adding properties like health points, levels, etc.

The Ai Behaviour Controller is a specialized reputation system as stated in Section 2, it is developed in C# Because we want it to work as an asset in Unity Engine,and it is focused on using a simple personality model to make the characters’ reactions to the stimuli in game believable and dynamic, the current version of the AI Behaviour controller does not take the sociology models into consideration yet.

Several experiments have been conducted with the system, a screenshot of the sample character class described in Figure.3 being created in the actual AI Behaviour controller has been given in Figure 4. It is found that the system is able to make different characters from the same class to have dynamic reactions to the same stimuli as shown in Figure 5. For example, in Figure 5, the “Guard 1” and “Guard 5” are the brave guards, when they detect the player, they will try to attack the player aggressively, and they will choose to fight back if they are attacked by the player first, meanwhile the other guards “Guard 2”, “Guard 3” and “Guard 4” are the timid guards, they will ignore the player when they detect the player, and when they have been attacked they will choose to run away rather than fight the player. Although all of the guards are sharing the same scripts and same decision tree, their reactions to the same stimulus solely depends on their own “personality” and can be completely different.

The above examples have shown that the AI Behaviour controller can make the characters using the same scripts and same animation controller to react differently to the same type of stimulus in the game world as we expected, although there are still much can be improved, but we can say this version of AI behaviour controller already has the basic features to be used as an improved version of the regular reputation systems.

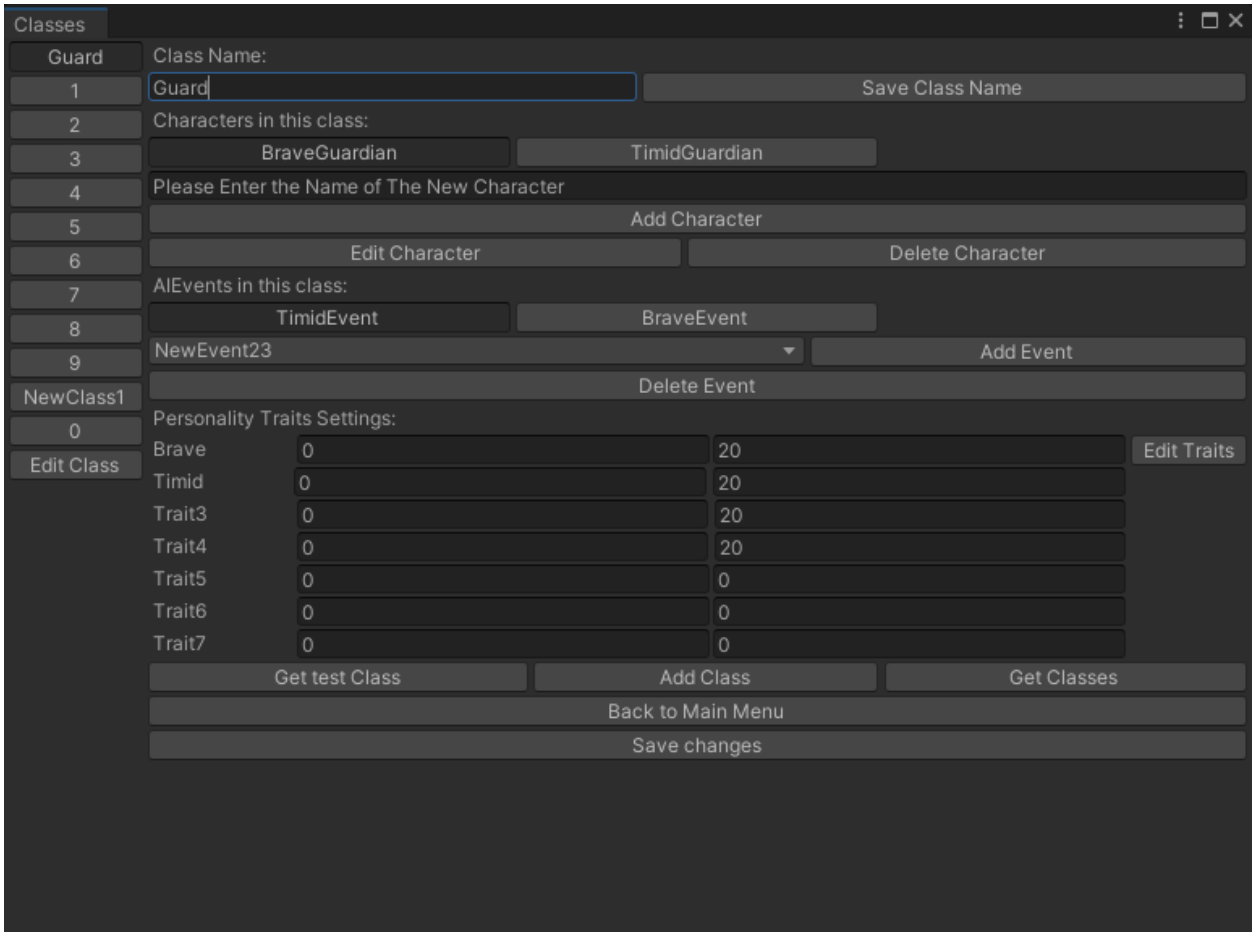

Figure 4. “Guard” class in AI Behaviour Controller

Figure 5. The record of the reactions of characters from the “Guard” class

5. Discussion

The current version of AI Event Controller in Unity has all of the functions we expect before the development, it is easy to learn and use, and saves a bunch of time for the game developers in managing characters and events in game. More importantly, it allows the game developers to design unique “personalities” for their game characters, and make those characters react believable and dynamic to the stimulus in the game world.

The limitations of the system are also obvious, first of all, the system does not consider the “relationship” of the characters yet, however, as stated in Section 2, the sociology model is also an important factor that needs to be considered to make the characters believable in game. Second, the weighted randomized formula currently used in the AI Event controller is inspired by the Reject Sampling, basically, after the weights of the events have been calculated, the system will create a list with events populated by their own weight, for example, if the weight of event A is 1, and weight of event B is 2, then the list will be {A,B,B}, then the system would choose a random number “i” between 0 and the length of the list, and the system would return the event in index “i” to the character. This method would have serious performance issues when there are too many characters who need to decide their reactions to the stimuli at the same time. As a result it is necessary to find a way to optimize the randomized system.

References

Rabin, S. (2002). *AI Game Programming Wisdom (Game Development Series)* (1st ed.). Charles River Media.

Byl, P. D. (2004). *Programming believable characters for computer games*. Charles River Media.

Brown, J., Lee, J., & Kraev, N. (2021). Reputation Systems for Non-Player Character Interactions Based on Player Actions. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *13*(1), 151-157. Retrieved from https://ojs.aaai.org/index.php/AIIDE/article/view/12950

Bailey, C., You, J., Acton, G., Rankin, A., & Katchabaw, M. (2012). Believability through psychosocial behaviour: Creating bots that are more engaging and entertaining. *Believable Bots*, 29–68. https://doi.org/10.1007/978-3-642-32323-2_2

Rosenthal, C., & Congdon, C. B. (2012). Personality profiles for generating believable bot behaviors. *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. https://doi.org/10.1109/cig.2012.6374147

Kersjes, H., & Spronck, P. (2016). Modeling believable game characters. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. https://doi.org/10.1109/cig.2016.7860412