Companion AI for Starbound Game Using Utility Theory

Elena Lebedeva
AI in Games Development Lab
Innopolis University
Innopolis, Russia
e.lebedeva@innopolis.ru

Joseph Alexander Brown
AI in Games Development Lab
Innopolis University
Innopolis, Russia
j.brown@innopolis.ru

***Abstract*—Starbound is a popular video game that has player companions as one of its features, but its AI lacks awareness about the player's state, the enemies' state and its own. This study aims to solve it by implementing a new companion AI which is based on utility theory. The AI was developed, and the user study was conducted to evaluate its ability to assist the player. The fighting ability was assessed by measuring AI's ability to stay alive during the battle and dealing damage to enemies without the player's interaction. As a result, the majority of user study participants did not notice the difference between the built-in AI and the one developed in this study. The participants that noticed the difference enjoyed each type of companion in equal measure. However, the fighting ability test showed that the utility-based AI stays alive longer and deals more damage compared to the built-in agent within the game currently.**

***Index Terms*—AI, Non-player Character, Video games, Utility theory**

I. Introduction

As you explore the world of *Starbound* [1] as the romantic space opera commander of a ship exploring the void, only to find yourself a cast way crash-landed on a distant planet. The player chooses one of the races for their character and then travels from one world to another, collects resources, fights monsters, and makes upgrades for their space ship. As you explore, you might need to make some friends, and the more intelligent they are, the better is your experience.

According to [2]: "AI can be applied to most aspects of game development and design including ... controlling the Non-Player Characters (NPCs)" (p. 1). NPC's primary purpose is to entertain the player of a game: e.g. by being an enemy that the player is required to eliminate. Warpeflet and Verhagen [3] provides a taxonomy of the roles of NPCs in games, and Warpefelt [4] examines further their believably. Another example is assisting the player by fighting with game enemies. The latter case will be observed in this study.

This type of NPC has existed in games for a long time: in *Half-Life* [5] the player was assisted by NPCs that were able to take simple commands "follow" and "stay", shoot at the enemies and give medicine if the player had a low health level. In *Fallout* [6] the player could also exchange supplies with their virtual companion. The more recent example could be *God of War* [7], where the player was followed by the companion that supported them during the battles in the game.

Starbound has a game mechanic that gives the player a possibility of hiring *Crew Member* NPCs for their ship. These NPCs can travel together with the player and either give a static effect (e.g. the increasing of the jump height or the boosting to the movement speed) or help during the battles with hostile NPCs. The fighting ability of Crew Member NPC has several problems:

1) **No consideration of the player's state.** The NPC's actions do not depend on the player's health. For instance, the NPC could come to the player and help them fight the monsters close to the player if the player's health is low.
2) **No consideration of own state.** The NPC does not choose the weapon depending on the distance to the enemy and does not try to keep distant from the hostile NPC if its health is low.
3) **No consideration of the enemy's state.** The NPC could perform some additional actions depending on the enemy's health, such as choosing an opponent among a group of enemies with the lowest health.

These problems can make Crew Member NPC an inefficient fighter and can cause a player's frustration. As a result, the decrease of a game experience quality occurs.

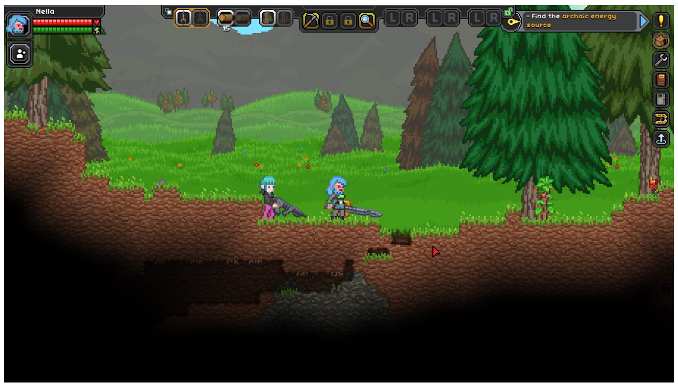

Fig. 1. A screenshot of Starbound gameplay. The player's avatar is in the center, their companion NPC is on the left.

978-1-7281-8763-1/20/$31.00 ©2020 IEEE

The remainder of this paper is organized as follows: Section II examines the related methods for creating helper NPCs; Section III contains the detailed description of the method used during the development of Crew Member NPC's AI; the experiment which is intended to test the developed in this study AI and its results are described in Section IV; conclusions made from the obtained results and proposed future improvements are described in Section VI.

II. Related Work

Many of the papers about companion NPC AI discuss the human-like AI for competitive multiplayer games. For example, the authors of [8] make a comparison on an evolving teammate bot that aims to score as many points as possible and die rarely and a hand-coded bot that aimed to be often seen by the player. They concluded that players prefer different bots depending on their expectations towards the gameplay.

An example of a study about companion AI in a single-player game could be [9]. The authors investigated a new approach to companion NPCs in the Third-Person Shooter game. They created an adaptive AI that changes its behaviour depending on game intensity. This AI was tested against non-adaptive companion and companion with adaptive firepower based on Dynamic Difficulty Adjustment (DDA). As a result, their approach compared to a non-adaptive companion demonstrates more respect for gameplay flow and provides an acceptable rate of game intensity level. Additionally, this approach could be used together with DDA. Gabele et al. . [10] examined representational aspects, but not mechanical elements of their NPCs to allow for a relation to form between a character and the player quickly.

In [11], Scott and Khosmood created a framework for a companion AI in a tower defence game. The companion was able to perform "complimentary" actions that are "...contrasted with a mimicking [12] action and is defined as any action by a friendly non-player character that furthers the player's strategy". To evaluate the performance of the companion, the authors conducted a user study, which showed that the majority of the participants found the companion useful.

In this paper, Utility theory is applied. This methodology allows us to create considerations that AI uses to determine which action is better to perform. These considerations could be used to solve the problems of Crew Member NPC's AI stated in section I. Utility theory application can be seen in [13]. The authors created utility-based AI for a mixed reality character to represent a resident of a small town in Iraq or Afghanistan. According to the users of the simulator with this character, it was close to people they would expect to see in these countries.

III. Proposed method

Utility theory [14] has the idea that AI has a set of actions to perform. Every time before picking an action, AI weights these actions by calculating the *utility function*. Utility function maps some parameters of the environment onto a value that describes how desirable the action is, typically a real number from 0 to 1. There are several actions that Crew Member NPC can perform (i.e. attack) and *decorators*. The decorator represents the factor that makes an action either more or less desirable as a *score*. In this case, a score is a real number from 0 to 1.

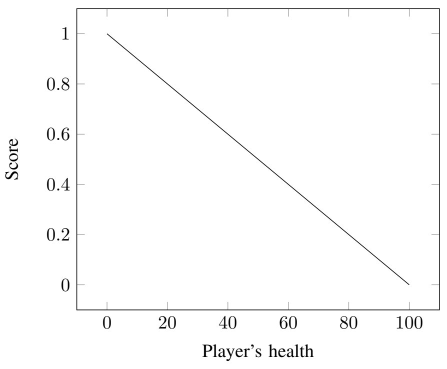

Fig. 2. Linear function for "Player's health is low" decorator

An example of a decorator could be "Player's health is low." decorator for the action "Protects the player". The decorator will use a decreasing function on the player's health value (a linear function in this implementation) as it is shown in Figure 2. The score is calculated as follows:

$$Score = -\frac{H}{H_{max}} + 1 \tag{1}$$

where H and H_{max} are the player's current and maximum health respectively.

The lower the player's health is, the higher is the score returned by the decorator is, therefore, the more the action "Protect the player" is desired to be performed.

An example of a non-linear utility function is the function used in "Health is high" decorator who is depicted in Figure 3. The quadratic function was chosen in such a way so the AI would react to the decrease of Crew Member NPC's health more desperately.

An action can have multiple decorators that form the utility function of the action:

$$Utility = \frac{\sum_{i=1}^{N} Score_i}{N} \tag{2}$$

where N is the total number of decorators.

The actions "melee attack" and "range attack" with the decorators "target is close" and "target is far" were designed in order to improve Crew Member NPC's approach to choosing the weapon: if the enemy is close to them Crew Member NPC will choose the melee weapon and the range weapon otherwise. In cases when Crew Member NPC did not have enough energy to use the ranged weapon, the " move closer" action will be triggered due to "not enough energy" decorator. This action should not be performed if Crew Member NPC's health is low, and because of that, it has "health is high" decorator, so the companion will not rush recklessly into the enemies. The action "come closer to the player." serves a

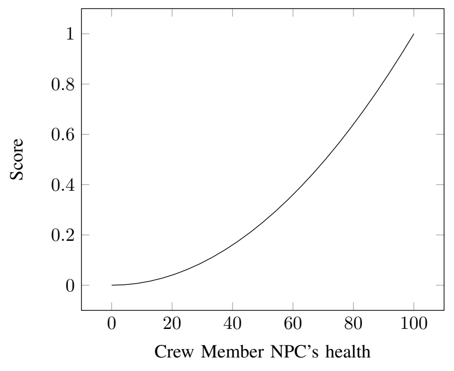

Fig. 3. Quadratic function for "Health is high" decorator

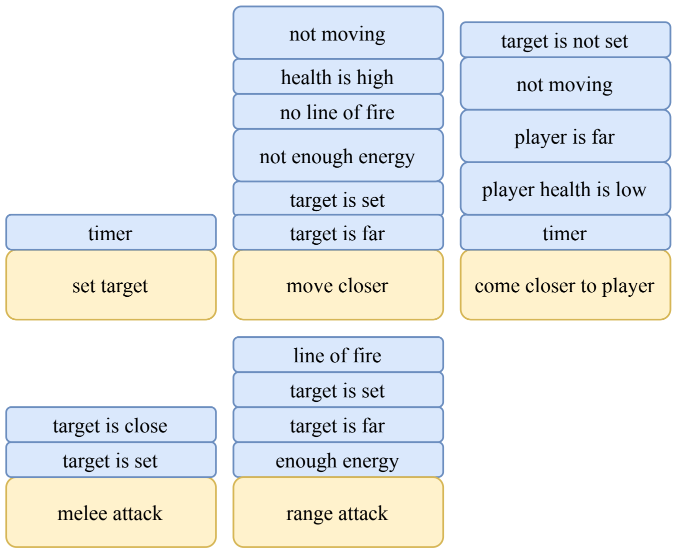

Fig. 4. Actions and corresponding decorators developed for Crew Member NPC's AI. Decorators are colored blue, actions are colored yellow.

purpose of supporting the player and has decorators "player is far," and "player health is low" for that. This approach was inspired by the AI that was focused on following the player [8].

The complete picture of Crew Member NPC's AI can be seen in Figure 4. Tables I and II contain descriptions for actions and decorators of the system. Table III contains utility functions that were used for the decorators.

TABLE I
LIST OF ACTIONS

Name	Description
set target	Set the nearest to Crew Member NPC entity as a target for attack. Iterate entities in a constant radius and ensure that the target is aggressive and alive.
move closer	Try to move closer to the target using pathfinding.
come closer to player	Try to move closer to the player using pathfinding.
melee attack	Use melee weapon.
range attack	Use range weapon.

Fig. 5. The monster that the subjects were required to meet during the experiment

IV. Experiment

In order to evaluate the quality of the developed Crew Member NPC's AI, the experiment with human subjects was done.

A. Experiment Protocol

The following steps describe the procedure of the experiment:

1) The subjects play the game where they are required to reach the monsters (see Figure 5) of a particular type while defending themselves from enemies with the Crew Member NPC assisting them. Each subject plays the game twice:
    a) with Crew Member NPC that has build-in AI, i.e. as it is presented the game;
    b) with Crew Member NPC that has utility-based AI developed during this study.
2) The subjects fill the form where they are asked if they noticed any difference. If the answer is yes, then they are asked which version of Crew Member NPC they would prefer. Also, the subjects were asked if they had any experience with *Starbound*.

Requirements and conditions:

TABLE II
LIST OF DECORATORS

Name	Description
timer	Circular timer that ticks after a constant period.
not moving	Crew Member NPC is moving.
health is high	Crew Member NPC's health is high.
no line of fire	The target is not on Crew Member NPC's line of fire.
not enough energy	Crew Member NPC does not have enough energy that is used for the range weapon.
target is set	Crew Member NPC's target is not empty and is valid.
target is far	Crew Member NPC's target is outside of the constant melee range.
target is not set	Crew Member NPC's target is empty or invalid.
player is far	Player is outside of the constant melee range.
player health is low	Player's health is low.
target is close	Crew Member NPC's target is inside the constant melee range.
line of fire	The target is on Crew Member NPC's line of fire.
enough energy	Crew Member NPC has enough energy that is used for the range weapon.

TABLE III
FUNCTIONS OF DECORATORS

Name	**Score**
timer	$S = \begin{cases} 0 & \Delta > p \\ (\frac{\Delta}{p})^4 & otherwise \end{cases}$ where p is a tick period and Δ is the time difference between ticks.
not moving	$S = \begin{cases} 1 & \lvert V \rvert > 1 \\ 1 - \lvert V \rvert & otherwise \end{cases}$ where V is a Crew Member NPC's movement velocity.
health is high	$S = (\frac{H}{H_{max}})^2$ where H is the current Crew Member NPC's health level and H_{max} is the maximum possible Crew Member NPC's health level.
no line of fire	$S = \begin{cases} 1 & C = true \\ 0 & otherwise \end{cases}$ where C is a predicate "there is a line collision between Crew Member NPC and its target"
not enough energy	$S = (\frac{E}{E_{max}})$ where E is the current Crew Member NPC's energy level and E_{max} is its maximum possible Crew Member NPC's energy level.
target is set	$S = \begin{cases} 1 & T = true \\ 0 & otherwise \end{cases}$ where T is a predicate "target is empty or invalid or dead"
target is far	inverse of "target is close"
target is not set	inverse of "target is set"
player is far	$S = \begin{cases} (\frac{d}{D})^4 & d < D \\ 1 & otherwise \end{cases}$ where d is the distance between the player and Crew Member NPC and D is a constant melee distance for Crew Member NPC
player health is low	$S = (\frac{H}{H_{max}})^2$ where H is the current player's health level and H_{max} is the maximum possible player's health level.
target is close	$S = \begin{cases} 1 & d < D \\ 0 & otherwise \end{cases}$ where d is the distance between the target and Crew Member NPC and D is a constant melee distance for Crew Member NPC
line of fire	inverse of "no line of fire"
enough energy	inverse of "not enough energy"

Fig. 6. The appearance of Crew Member NPC during the experiment

TABLE IV
USER PREFERENCES ON COMPANION AIs

	Had experience with the game	Played the game for the first time
Did not notice the difference	2	7
Preferred utility-based AI	2	1
Preferred built-in AI	2	1

- The game is run on the same personal computer in order to prevent technical issues and differences between machine capabilities from influencing the subject's opinion.
- The order in which two versions of Crew Member NPC's AI is presented to the subject is random to minimize the influence of the subject's skill difference during each play, in their opinion.
- The subjects are not told before the experiment what they are going to judge in the game.
- The look of both Crew Member NPC versions is the same (see Figure 6).

B. Results

15 subjects participated in the experiment. 6 of them had experience with *Starbound*, while the others played this game for the first time. 6 subjects noticed the difference in Crew Member NPC between each of the two plays. 3 of them said that they prefer the utility-based version of Crew Member NPC's AI; the rest of them said the opposite. Table IV shows the detailed picture of user preferences.

The subjects that chose the built-in AI said that it was less aggressive and did not attract attention to the enemies as much as another companion did. The counterpart explained their choice by saying that the new companion was more effective: it followed the weapon of their choice , and its attacks were more effective. Both groups stated that the built-in companion was always trying to keep closer to the player, and the other one was not.

V. Performance Evaluation

To test the fighting ability of Crew Member NPC's AI, built-in and utility-based AIs were put on an arena level, where they were required to kill as many enemies as they can before they die without the player's interaction.

TABLE V
PERFORMANCE EVALUATION RESULTS

	T, seconds		Damage	
	utility-based	built-in	utility-based	built-in
average	15.367	11.967	0.124	0.034
best	29.784	24.596	0.255	0.074
worst	8.56	8.412	0.022	0.003
σ	5.352	3.229	0.065	0.018
p-value		0.004		0.000

Each of AIs was tested 30 times. During the test the time T for which the companion stayed alive, and *Damage* were measured. Damage is calculated as follows:

$$D = \frac{\sum_{i=1}^{N} P_i}{N} \tag{3}$$

where N is the number of enemies on the level and

$$P_i = 1 - \frac{h_i}{H_i} \tag{4}$$

where h_i is the ith enemy's current health and H_i is the enemy's maximum health.

The results of the test can be seen in Table V. Damage here is calculated after Crew Member NPC's death.

In order to show that the differences between the bots were not due to random chance, but due to the behaviour characteristics of the agents, a two-tailed t-test was performed. The p-values from a two-tailed t-test show that the differences are statistically significant with an extremely low margin of potential error. This proves that the utility-based agent has categorically better damage output and will survive longer.

VI. CONCLUSIONS AND FUTURE WORK

During the experiment the subjects that had no experience with *Starbound* were noticeably struggling with the game controls and mechanics. Because of that they were paying less attention to Crew Member NPC's behavior. Therefore in order to fully test Crew Member NPC's AI it is required to run the experiment with subjects that are familiar with the game, e.g. members of *Starbound* online community. This would make the experiment more focused on the target audience of the game for which Crew Member NPC's AI can be important. It is also because different players may prefer companions with different types of behaviour: some of the subjects described the utility-based companion as "tank" and the built-in one as "support". "Tank" is a common term for a playstyle in which the player actively uses melee attacks against the enemies performing a role of a shield, while "support" player attacks from a distance. [15]

The performance testing showed that, on average, the utility-based AI stays alive for a longer time and deals more damage to enemies.

Since the AI's utility system was developed as a modification to the game using the provided modification API and was written in the interpreted language *Lua* [16], the developed system may have performance issues compared to the system that is built in the game engine of *Starbound*.

As a further improvement, the actions of the utility system could be separated into two categories: movement and attack. The process of choosing activities from these categories could be concurrent that will allow the AI to move and attack simultaneously. During the human testing, it was determined that when the player's health is close to 50% or less, AI will only pick "come closer to player" action and do not attack enemies. Concurrent systems can solve this issue.

Also, a possible improvement could be a more intelligent choice of the target, which can solve the cases when AI attracts the enemy's attention, when the player does desire this outcome. Player's focus and enemies' health could be taken into consideration by AI when choosing the target for attack.

Overall, the players do not always pay attention to their companions in Starbound. When they do, they might prefer a more cautious and passive companion even though its intelligence might lower.

REFERENCES

[1] Chucklefish, *Starbound*. Chucklefish, 2016.

[2] S. M. Lucas, "Computational intelligence and ai in games: a new ieee transactions," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 1–3, 2009.

[3] H. Warpefelt and H. Verhagen, "Towards an updated typology of non-player character roles," in *Proceedings of the International Conference on Game and Entertainment Technologies*, 2015.

[4] H. Warpefelt, "The non-player character : Exploring the believability of npc presentation and behavior," 2016.

[5] Valve, *Half-Life*. Valve, 1998.

[6] B. I. Studios, *Fallout*. Interplay Entertainment, 1997.

[7] S. M. Studio, *God of War*. Sony Interactive Entertainment, 2018.

[8] A. Friedman and J. Schrum, "Desirable behaviors for companion bots in first-person shooters," 2019.

[9] J. Tremblay and C. Verbrugge, "Adaptive companions in fps games." *FDG*, vol. 13, pp. 229–236, 2013.

[10] M. Gabele, A. Thoms, J. Alpers, S. Hußlein, and C. Hansen, "Non-player character as a companion in cognitive rehabilitation for adults - characteristics and representation," in *Proceedings of the 3rd International GamiFIN Conference, Levi, Finland, April 8-10, 2019*, ser. CEUR Workshop Proceedings, J. Koivisto and J. Hamari, Eds., vol. 2359. CEUR-WS.org, 2019, pp. 130–141. [Online]. Available: http://ceur-ws.org/Vol-2359/paper12.pdf

[11] G. Scott and F. Khosmood, "A framework for complementary companion character behavior in video games," *arXiv preprint arXiv:1808.09079*, 2018.

[12] T. Angevine, "Mimica: A general framework for self-learning companion ai behavior," 2016.

[13] K. Dill and L. Martin, "A game ai approach to autonomous control of virtual characters," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2011.

[14] S. Rabin, *Game AI Pro: Collected Wisdom of Game AI Professionals*. A K Peters/CRC Press, 2013.

[15] J. Kim, B. C. Keegan, S. Park, and A. Oh, "The proficiency-congruency dilemma: Virtual team design and performance in multiplayer online games," in *Proceedings of the 2016 CHI conference on human factors in computing systems*, 2016, pp. 4351–4365.

[16] "Lua - about." [Online]. Available: https://www.lua.org/about.html