Species Autonomous Evolution in Games Using Player Behavior Modeling

Sidney Reis Lucas Fialho Soraia Raupp Musse*

Pontifícia Universidade Católica do Rio Grande do Sul, Programa de Pós Graduação em Ciência da Computação, Brasil

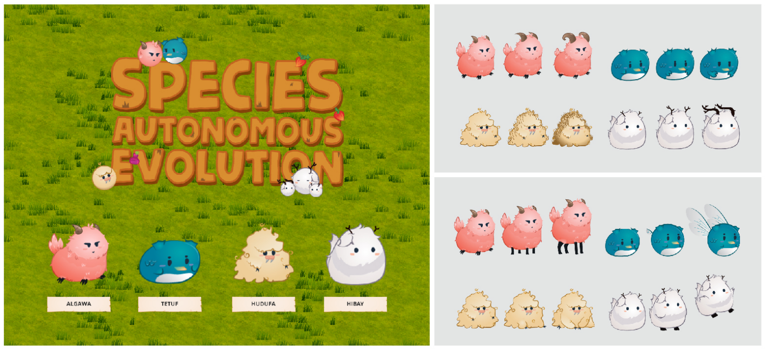

Figure 1: On the left: The game initial screen. On the right/up: the characters evolution based on defense attribute and, on the right/down, their evolution based on motion speed attribute.

Abstract

The usage of player modeling in games is constantly growing and being studied nowadays. It is a way of connecting the user to the game in a personal way, since the game will have different results and obstacles based on the user playstyle. Generally, games that use player modeling comes with a confirmation from the user to proceed with the changes in the game. Our proposed model is an attempt to make the game even more personal. We built a prototype that does not ask the player for guidance, the game autonomously change and shape the player characters by the way he/she acts in the game. The developed prototype is a species evolution game, where the player chooses one species from four available. This species is only controlled by the player, while the others are autonomously controlled by AI agents. The objective is to evolve and survive in the game, since there are some threats to the user like species combat and starvation, for instance. The results of this prototype was satisfactory, it was possible to create a game where the player will just play accordingly to their preferences, leaving the customization of the species to the player modeling automatically generate.

Keywords: autonomous evolution, games, player behavior, player model, playstyle, character customization

1 Introduction

The game industry is a challenging and constantly growing area. It requires a lot of effort from different other fields that begins from software development and can extend to psychology studies, for example. A game needs to be aware of the new hardware technologies to accommodate the game for the latest updates and also, it needs to be original or implement a different gameplay to make success [6].

Games where the player choices matters are becoming a very popular genre lately. This kind of game is made of choices that are selected by the user and it changes the process and sometimes the narrative of the game. These user preferences can be explicitly seen on games such as the Infamous (series) [7], where the character and its actions can be either evil or good. Also, an already known and still ongoing trend is games whose features allow players to modify characters body and their specific characteristics. Some of them have used these features in order to simulate their evolutionary behavior. These can be found implemented in games like Spore [1] and Niche [5], for instance. In Spore, the player is capable of modifying the species he/she controls, adapting their bodies and skills to fit the player goals. In Niche [5], it is possible to see a more complex evolution process based on the species genetics.

In this paper, we propose a game that learns the player playstyle to generate autonomous evolution of species. In this way, the evolution method should provide modifications in the species' characteristics w.r.t. player's actions. It is also important to notice that the player is able to control all characters of a chosen species, one at each time. With that being said, the characters not being controlled by the player are autonomous, making the game's world not completely user-dependent. In addition, such characters are not impacted by gamer decisions.

2 Related Work

This section presents the related work that contribute to the Player Modeling feature designed in the proposed method. Its goal is to personalize the user characters w.r.t. user actions. Missura and Grtner [3] proposed solutions for automatically adjust the difficulty of computer games for different types of players and supervised predictions from short traces of gameplay. The difficulty is increased overtime based on the ability, experience, perception and learning curve of each individual player. The hard part of this intelligent difficulty adjustment is the wrong choices, which can easily lead to players stopping to play the game if they get bored (for being too easy) or get frustrated (for being too hard).

Nery, Teixeira, Silva and Veloso proposed, in their work [4], a World of Warcraft player categorization based on the players behaviors through semi-supervised learning. Depending on the player's gameplay style, different categorizations can be applied. For instance, a player who prefers to interact with the world will have an Explorer's profile, but if the same is more focused on its character

*e-mail: soraia.musse@pucrs.br

(and continues interacting more than acting), it will be placed on the Socializer category.

Concerning games that approach the evolution area, in [2], Izidoro, Castro and Loula present the modeling of the population dynamics and biology evolution, aspects implemented in the Calangos, an educational game based on the Brazilian Caatinga fauna and flora. In addition, the authors proposed an application of those concepts in a simulator. In Calangos, the player controls the behavior of lizards found in Caatinga, affecting the ecology and evolution of those lizards.

3 Proposed Method

This section describes the proposed model used in our prototype. Firstly, it is important to notice that our model contains different entities impacting the whole game: the user, the autonomous and controlled characters and the generated newborn characters. Basically, while the user choses and controls only one of the species (formed by creatures that can one at each time be controlled as well), the other species are autonomously animated. For all creatures, attributes and actions were defined in order to evolve in the game, as described in next sections.

The attributes affect the actions applied by creatures in the game. The actions are what the creatures are able to do in the game. Actions can be autonomously performed by autonomous creatures or manually made by the player, when controlling one character. When applied, actions can modify attributes. For instance, an attack applied by a creature uses its attribute attack power and cause damage to the opponent's health points. Actions are detailed in Section 3.2. As said before, the role of the player in this game is not only interact in the virtual world, but feed the player's model with information about applied actions. These data is used in the evolution procedure modifying the attributes of newborn creatures. More details about the player model and evolution are described in Section 3.3.

3.1 Attributes

Attributes are the variables that represent creature states. It is directly influenced by the evolution, the player's actions and also autonomous actions, and defines how the species looks to the player (their visual behavior). Following we describe in further details the attributes for a character i. In addition, all these definitions are applied for autonomous and controlled creatures.

1. **Fatigue Level** (f): Creatures have a fatigue level (f). f_i value from creature i is an integer value in the interval $[0;MaxF]$. It is increased each frame the creature is moving and decreased each frame the creature is not moving. This attribute is in the interval $[0;3000]$, where 3000 is the maximum value $MaxF$. Associated to this attribute, two states are possible: *i) - Resting* is activated when $f_i = \frac{MaxF}{2}$ and deactivated when $f_i = \frac{MaxF}{15}$; and *ii) - Fast Resting* determines a more urgent need for rest and it is activated when $f_i = MaxF$ and deactivated when $f_i = \frac{MaxF}{1.5}$.

2. **Movement Speed** (s)

   The creature speed (s_i) defines how many space units a creature moves per second or frame. Important to note that in this proposed model we use the space unit adopted by the Unity game engine. The speed takes in consideration the upgrades in movement action that a newborn creature (affected by evolution) may have, meaning that after some evolution the creature could have some speed benefits. The evolution definition are detailed in Section 3.3. The attribute s is a positive rational number contained in interval $[0.1;0.2]$ and the value for initialization is 0.1.

3. **Health points** (h)

   Health points (also called *life* in the prototype) is the attribute that handles how much damage the creature can take before dying. When the health points $h_i = 0$ of creature i, the creature is permanently removed from the game. h is an integer number in the interval $[0;100]$ and the creatures are created with $h = 100$. h can be decreased by combat (attacks from enemy species) and starvation when the species does not eat in a long time. The attributes Attack and Starvation detail how these features impact on h attribute.

4. **Starvation Level** (t)

   Creatures only feed or search for food autonomously when they are hungry, which in this prototype means when the starvation level ($t = \frac{MaxT}{2}$), i.e. reaches the half of maximum level (t is in the interval $[0;MaxT]$), where $MaxT = 300$. In the prototype, a creature is hungrier when t_i is getting smaller. When $t_i = 150$, the creature starts searching for food autonomously (if the creature is not controlled by the player). Each fruit eaten by the creature makes $t_i = t_i + 200$, not overcoming the maximum value. When $t_i = 0$ the creature enters in the starving state. During this state, it loses 1 health points ($h_i = h_i - 1$) per second. Considering that the creature i starts with $h_i = 100$ in this prototype, it would take 100 seconds to die after reaching the starving state. To avoid it, the creature needs to eat fruits, either autonomously or controlled by the user.

5. **Perception Radius** (R)

   Each creature has a radius sight that influences how far they can perceive food when being autonomously controlled. It makes fruits, in the map, visible to the creatures, so they can eat. Indeed, creatures with larger radius of perception are able to find food more easily, since it can detect fruits further. The Perception radius (R) is defined in space units. The default value is $R = 180$ units for all creatures, which can be increased in newborn creatures caused by evolution, as can be seen later in Section 3.3. In Figure 2, we show two autonomous creatures (identified as numbers 1 and 2). The circular regions illustrate perception area for each one.

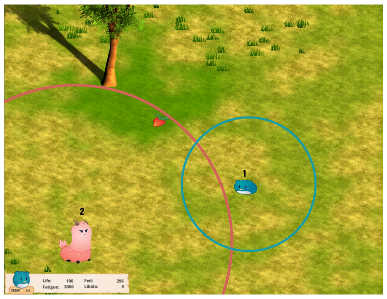

Figure 2: Perception radius range. Both creatures are hungry, but one of them is capable of seeing the fruit.

6. **Attack and defense power** (A and D)

   These attributes are responsible for attacking and defending activities of the creatures. The attack action (Section 3.2.2) uses attack and defense power to define how much damage is done to the attacked creature. These attributes are A and D, respectively representing attack and defense attributes. They are defined in the interval $[0;2]$ and initialized as 0 for all creatures. In addition, these values can be increased in newborn creatures, w.r.t. the evolution method, as can be see in Section 3.3.

7. **Libido** (l)

   As going to be mentioned in Section 3.2.3, the libido is used to define when the creature can breed. The attribute is set $l = 0$ for all creatures in the beginning of the game or when a creature is born, and in both cases it is increased by 1 at each second. It is in the interval $[0;200]$ and decreases by 50 each time the creature mates.

3.2 Actions

The actions can be autonomous and/or manually performed by the player and have outcomes in the game state, using creatures attributes as parameters and modifying those attributes.

3.2.1 Movement of Creatures

The main action of the game is the movement. Creatures can move across the map and this is necessary to make other actions. The player can move only one creature of chosen species at a time, but the rest of the creatures in the game, including the enemy ones, move autonomously across the map based on the same rules.

3.2.2 Attack

As in a wild environment, the prototype has the option to attack other species creatures, inflicting damage to their health points. This attack can be either manual (user action) or autonomous. When the player's current creature is attacked by an autonomous enemy creature, two new actions are possible: the action of escaping from the battle (Run from attacker) or attacking back the enemy (Counterattack). In this case, our model records the player actions in order to save the applied playstyle. It is going to be used in evolution method, as shown in Section 3.3.

3.2.3 Breed

The key action to enable autonomous evolution is the mating action. This is an action only triggered by the manual control from the player, so autonomous creatures can not mate in the current version of our game. When the player makes two creatures mate using the game interface, the breed happens and a newborn is generated (detailed in Section 3.3).

3.2.4 Feed

Creatures are able to both manually and autonomously feed themselves. Fruits are spread across the map, and those fruits are available to be eaten by the creatures.

3.3 Player Model and Evolution

The evolution process is the key part of the game. It uses some of the attributes information updated based on the user model. The goal is to represent the user's playstyle. The method developed is a rule-based system implemented through decision-tree. As presented in previous sections, the characters have attributes and some of them are used in the player model, as follows: *i)* Perception radius R, *ii)* Movement speed s, *iii)* Attack power A, and *iv)* Defense power D.

We implemented a histogram of actions in order to represent the *playstyle* of the user. In this way, the user playstyle is defined by how many times the user did each of main actions. For example, let us take the example of a user playstyle more focused on defensive than attacking. Indeed, when the player's creature runs away from attackers, counterattack or attack, the action counters are increased. When the player makes two creatures breed, these variables are checked in order to find out the most used action. So, the newborn creature will be impacted by this.

At the end of the evolution process, the newborn has its new attributes defined as presented in last section. Each of these updated attributes modifies certain aspects of the creatures and are visually illustrated in the game. The visual aspect is shown in sprites that are selected depending on the evolved attributes. For example, players that used to ran away from battles, will generate newborns with longer legs, i.e. improving their ability to run (shown in Figure 4). Players that counterattacked other species will create newborns with more tendency to protective behavior, so they will have artifacts to defend themselves (illustrated in Figure 5). To achieve such body changing feature, the creatures are represented in 2D form using sprites in the prototype. Figure 3 illustrates four possible visual behaviors for two different attributes.

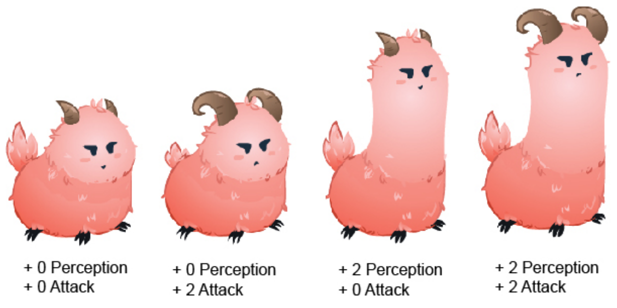

Figure 3: Visual representation of two creatures after evolving perception and/or attack.

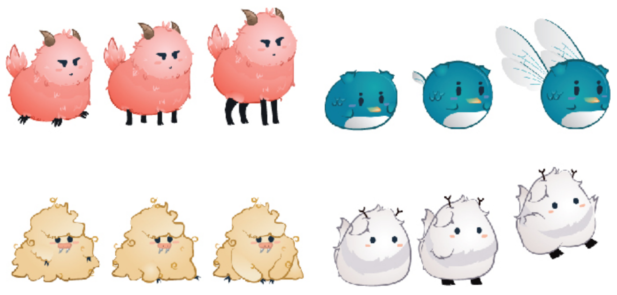

Figure 4: Visual representation of movement speed attribute evolution.

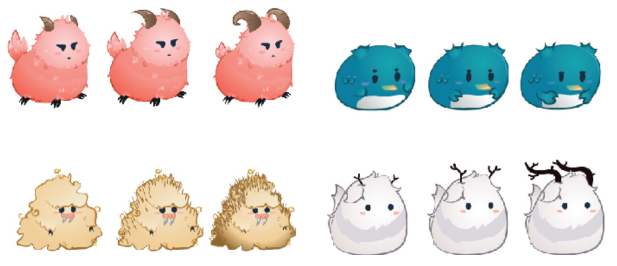

Figure 5: Visual representation of defense attribute evolution.

4 Experiments and Results

This section details the results found after testing the prototype with different mindsets accordingly to test cases that aims to validate the proposed model. In the test cases, we focus on testing key points in the game which the goal was to verify if the prototype works as a species autonomous evolution simulator, being influenced by the user playstyle.

Table 1: Test Cases summary.

	Current Model Before Test	Legacy Model Before Test	Parent Attributes	Actions	Newborn Attributes
TC1	Attacked: 0	Attacked: 0	Attack upgrade: 0	Attack all enemies encountered	Attack upgrade: 1
TC2	Foods eaten: 0	Foods eaten: 2	Perception upgrade: 1	Search for and eat fruits when not hungry	Perception upgrade: 2
TC3	Counterattack: 0, Ran away: 0	Counterattack: 2, Ran away: 0	Defense upgrade: 1, Movement speed upgrade: 0	Ran away from 6 of 6 incoming attacks	Defense upgrade: 0, Movement speed upgrade: 1
TC4	Counterattack: 0, Ran away: 0	Counterattack: 0, Ran away: 3	Defense upgrade: 0, Movement speed upgrade: 2	Counterattack 6 of 6 incoming attacks	Defense upgrade: 0, Movement speed upgrade: 1
TC5	Foods eaten: 0, Ran away: 0	Foods eaten: 0, Ran away: 0	Perception upgrade: 0, Movement speed upgrade: 0	Search for and eat fruits when not hungry, ran away from all enemy encounters	Perception upgrade: 1, Movement speed upgrade: 1

Details about the player actions are logged into text files each time the player performs any pre-defined action that can modify the playstyle model. With that in mind, as mentioned before, the four actions that trigger logs are: *i)* Player eats without the current controlled creature being hungry; *ii)* Attack nearby enemies; *iii)* Counterattack incoming attacks; and *iv)* Run away from battles.

The test cases used to verify our proposed model are following presented. Table 1 briefly summarizes the five test cases used in this analysis. It easy to perceive that the newborn attributes follows the player model. For instance, in test case 1 (TC1) the player showed a tendency to attack (see Action in 4^{th} column of Table 1. As a consequence, the newborn had an upgrade in Attack attribute. In TC2 we can see the perception attribute upgraded since the player chosen for search fruits and eat them even when the avatar was not hungry. In TC3 the player ran away from 6 incoming attacks, consequently the newborn downgrade the defense attribute (in comparison to the its parent) and upgraded movement attribute. In TC4, the newborn decreased the movement speed attributed as a consequence of counterattacking others and finally, in TC5 the player searched for fruits when the avatar was not hungry and also ran away from all enemies, as a consequence the newborn upgraded the attributes perception and movement.

Some illustrations of the game are shown in next figures. The combat is illustrated in Figure 6) If the current creature is within the attack range, the *Attack* button will trigger an attack, dealing damage and displaying an attack animation. It is also possible to open a breed menu (Figure 7).

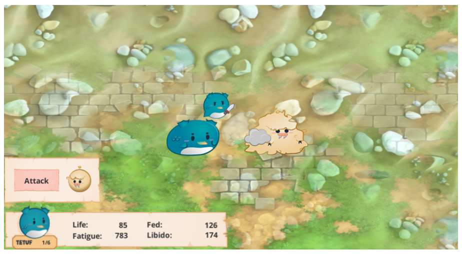

Figure 6: Offense menu and the attack animation.

5 Final Considerations

Even though the gaming field is very explored nowadays, the game development process needs hard work and time. In our view, the results of the prototype were considered satisfactory. It was possible to create a game where the player will just play accordingly to their preferences, leaving the customization of the species to the automatically generated player model. The game logs show that the current prototype state and the proposed model are capable of generating creatures that resembles the user playstyle. As possible future work, we consider that group behavior and autonomous creatures breed and evolution would be the main features to be developed. The group behavior feature would make creatures walk together when near to each other and attack other creatures together. Also, to make the game more challenging and natural, the autonomous creatures breed and evolution would make enemies breed and evolve just like the player.

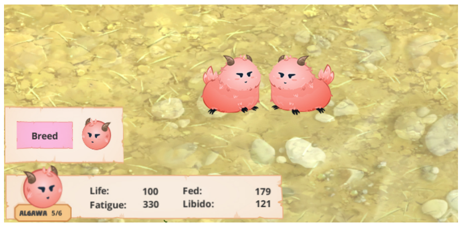

Figure 7: Breed menu.

References

[1] E. A. Inc. Spore, 2009.

[2] V. N. L. Izidoro, L. N. d. Castro, and A. C. Loula. A genetic-evolutionary model to simulate population dynamics in the calangos game. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 277–283. IEEE Computer Society, Out 2010.

[3] A. Missura and T. Grtner. Player modeling for intelligent difficulty adjustment. *LeGo-09*, 2009.

[4] M. S. Nery, R. A. S. Teixeira, V. d. N. Silva, and A. A. Veloso. Setting players' behavior in world of warcraft through semi-supervised learning. In *SBC - Proceedings of SBGames 2013*, pages 221–228. SBGames, Oct 2013.

[5] T. Niche. Niche a genetics survival game, 2016.

[6] M. W. Peter Zackariasson and T. L. Wilson. Management of creativity in video game development. *Services Marketing Quarterly*, 27(4):73–97, 2008.

[7] S. P. Productions. Infamous (series), 2009.