Genetic Algorithm in Survival Shooter Games NPCs

Alisson Steffens Henrique
Universidade do Vale de Itajaí
Itajaí, SC, Brasil
ali.steffens@gmail.com

Rudimar Luis Scaranto Dazzi
Universidade do Vale de Itajaí
Itajaí, SC, Brasil
rudimar@univali.br

Ricardo Martins Brasil Soares
Universidade do Vale de Itajaí
Itajaí, SC, Brasil
r3soares@gmail.com

Rodrigo Lyra
Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil
rodrily@gmail.com

ABSTRACT

Games must engage players by keeping them in the game flow. To better define the game difficulty according to the player, Genetic Algorithms can be used. One of the interesting characteristics of Genetic Algorithm is that it is a non-deterministic algorithm. For the player's vision, it means that enemies are unpredictable. By not knowing which NPCs he will face, the gameplay turns more interesting. Another amusing factor for gaming is its adaptability, causing NPCs to slowly struggle to find a way to beat the player. These two characteristics make Genetic Algorithms good tools to make games more entertaining. This paper aims to demonstrate this adaptation capability in the Survival Shooter, developed by Unity enterprise and modified by the author for the algorithm implementation. As result, it shows that players could stay in the game flow while playing against genetically modified enemies.

KEYWORDS

survival shooter, Genetic Algorithms, Unity, game, adaptation, Artificial Intelligence, NPC

1 INTRODUCTION

Artificial intelligence has been present in digital games development since - at least - the 40's. At this time a mathematician called Raymond Redheffer developed a computerized version of Nim, in which an artificial intelligence played against the player [8]. Those days, the computer used to do only mathematical instructions and discrete logic, but these first NPC were very important in the evolution of games for the development of new Artificial Intelligence.

Evolutionary Computing is a part of AI in which computers aim to emulate nature's evolutionary behavior. When an intention to carry out a simulation of it, Genetic Algorithms are usually the answer. They are based on the natural selection for species' adaptation that is used in its execution. Its operation follows around some Darwinist concepts focused on the evolution and environment adaptation theory [2].

The purpose of these algorithms is to be adapted to the environment until it is possible to survive the competition within it. This evolution or adaptation of the individuals is given through the occurrence of crossovers and mutations amongst the population, which in turn is evaluated according to its fitting to the environment [1].

This paper demonstrates the effectiveness of this technique in survival games where the player fights a population of Non-Playable Characters (NPCs) which are controlled by the computer. As the main topic of this paper, a modified version of the Survival Shooter game, developed by Unity Technologies, was used.

In this game, the character - controlled by the player - is placed in an environment (map), where he must survive to countless hordes of enemies fighting him. The player's score is calculated by the number of hordes he survives (by eliminating all enemies), and the time he takes to achieve these goals.

Each enemy has the ability to make decisions regarding their actions and moves. These decisions (defined in their DNA here represented as parameters) are based on information related to the environment in which the character is located. The positions and actions of the players and their enemies', as well as the character's intrinsic attributes, such as its current health points or items, are available for usage.

While playing, these individuals are able to make decisions based on the moment they are "living". They can fight the player by using different weapons and abilities or interact with other NPCs for support. In addition, he may choose to run away from the player by reaching a critical damage status.

Due to the possibility of configuring several parameters for the algorithm, many comparisons were made between these possibilities in order to obtain the best individual's performance in the game. Therefore, the performance of the algorithm was tested by using real players.

2 Genetic Algorithms

The first computational model of Genetic Algorithms was developed by John Holland [4], who at the time, did a study on the evolutionary processes of nature [5]. This model, described in Figure 1, can be divided into four stages: Initial Population, Selection, Crossover and Mutation. After the creation of the new population, the cycle starts over again.

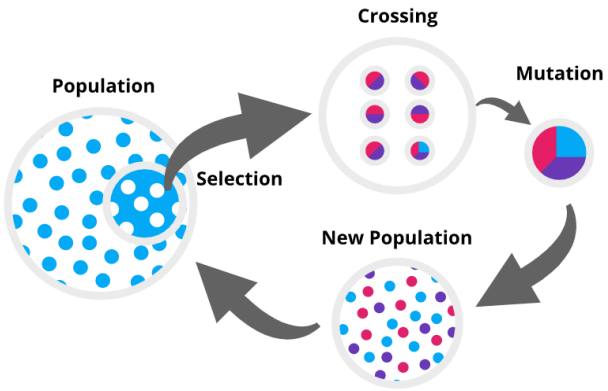

Figure 1: Genetic Algorithm Steps

2.1 Former Population

The creation of the initial population is based on random chromosomes, seeking to fill the search space as widely as possible. This population must be, among all the others, the most heterogeneous.

When creating a new individual, its genes are defined as presented in Table 1.

Gene	Values	Description
Appearance	0-2	bunny, bear or elephant
Weapon	0-4	punch, flamethrower, missile launch, bomb, thunder
Range	1.5-15	distance between the NPC and its target
Health Points	30-500	character's life
Damage	1-20	character's attack damage
Attack Speed	0.5-2	The lower the value, the faster the attack
Movement Speed	1-5	The reference value is 3 (player speed)
Healer	0;1	1 is healer, 0 is not
Run away	0;1	1 run away with 25% of health
Accuracy	1-100	chance to hit target

Table 1: Genes

In games, in order to guarantee the diversity of the population, it is common to select an initial population beforehand, ensuring that it spreads satisfactorily through the search spectrum of the algorithm. So players do not get frustrated in the first hordes, in addition to allowing a greater chance of adaptation to the player's game method [6].

2.2 Selection

For adapted individuals' selection to compose the new population, three main methods are used: tournament, roulette, and cut. This selection is related to the survival concepts of the fittest subjects, described by the theory of evolution [2,3].

The chosen selection algorithm strongly impacts the application's final result. As this paper aims to do a genetic algorithms' study in games, all of the selection methods were applied, to make it possible to choose the best one before the game begins.

Table 2 shows a comparison between the selection methods used (applied in this paper), as well as the way in which each one impacts the final result of the populations.

Method	Features
Cut	High selective pressure, only the best individuals will be selected
Roulette	Low selective pressure, allows individuals with low fitness to be selected
Tournament	The variation between previous methods

Table 2: Selection Methods

The game in discussion allows - when selecting the total population size - to define the method of selection and cut range (how many will be selected), which is usually given by a quarter of the initial amount.

To define each individual's score, a subject's fitness test is used. This test is calculated just after the death of each NPC. This calculation, in turn, is based on three main health-related characteristics of the individual, which are expressed in Table 3.

Metrics	Default	Requirements
Survival	10	Escaping and being healed by an ally
Damage Health	0,4	For every point of damage/healing dealt with the target
Time Alive	0,2	For every second lived

Table 3: Metrics for testing

These points are used as the basis for the fitness calculation, which will be given by a normalized value and distributed in the interval between the real numbers 0.9 and 1.1 [5]. The normalization of this fitness is described by the equation:

$$E(i,t) = Min + (Max - Min) * \frac{rank(i,t) - 1}{N - 1}$$

Equation 1: Fitness Standardization

Where: (I) Min is the value of the evaluation that will be assigned to the worst ranked individual; (II) Max is the value of the evaluation that will be assigned to the best-ranked individual; (III) N is the number of individuals in the population; and (IV) Rank (i, t) is the ranking of individual i in the population kept by the algorithm in generation t.

2.3 Crossover

After the best individuals of the population have been selected, the crossover process begins. In this stage the exchange of genes occurs between the selected individuals, seeking to generate new individuals that are variations of the winners.

There are two most common methods for this cross: Simple Arrangement and Same Species. The arrangement method consists of crossing the individuals two by two, based on the order in which they appear. This simple crossover between the parents is based on the exchange of chromosomes for the generation of DNA, as it happens with living beings. An example of its operation in the game can be seen in Figure 2.

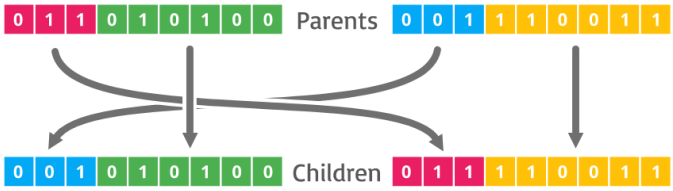

Figure 2: Simple Crossover

The number of individuals generated by this technique grows exponentially as shown in Equation 2, where n is the number of individuals selected. Therefore, in cases where the selected population is very large, it was chosen to eliminate surplus individuals in a random manner.

$$A_{n,2} = n^2 - n$$

Equation 2: Number of individuals

Much like the Simple Arrangement method, the Same Species method adds a limitation to these arrangements, making only individuals with the same appearance gene to crossover. In this mode, it is frequent the occurrence of individuals without pair, which in turn results in the death of the same and disappearance of the species. For this reason, the method is considered ineffective in experiments where the selected population is small.

2.4 Mutation

After crossover, there is a chance whereas the new individual undergoes a mutation. If this happens, he will have new information, which does not come from any of its parents. This mutation occurs in a random manner and it must have a low probability, since otherwise, the behavior of the population will become very random and non-adaptive [5].

The initial mutation rate was 5%, but two rules were implemented that could change it during execution. These rules have as main factors of influence the difficulty that the player demonstrates, and the domination of some species.

The difficulty is set based on the average damage taken by the player during the last 3 rounds. If this damage is less than 25% of the player's life, then it is considered that the hordes are very easy. If that percentage reaches above 75%, it is considered difficult. And values between 25% and 75% are considered average.

Depending on the options chosen by the player, it is possible to increase or decrease the mutation rate according to the relative difficulty to generate more heterogeneous populations.

The domination of a species occurs when more than 80% of the individuals have same appearance (Rabbit, Bear or Elephant). This indicates a convergence of the algorithm and that the evolution of individuals tends to stabilize. In this case, it is also possible to temporarily increase the mutation rate of this gene to thereby obtain individuals of different species.

The system still allows the player to choose the number of genes that will be modified if a mutation occurs, allowing mutations to generate higher or lower genetic variability.

3 Survival Shooter

The game referred by this article was developed by Unity company and consists of a character controlled by the player who must survive to endless hordes of enemies. To defend itself, the player relies on a machine gun with unlimited ammunition. The game environment consists of a children's room in which the character, who is a baby, must fight against rabbits, bears and plush elephants that attack him constantly. The game ends only after the player's death.

The original version was modified to allow the inclusion of the Genetic Algorithm and increase the variety of enemies. Figure 3 shows the original version.

Figure 3: Survival Shooter original version

4 Performance

In order to analyze the performance of the algorithm, different configurations were defined to find the one that best suits the game. Table 4 shows some of the settings applied in the algorithm, relating the selection method to the population, and showing the average fitness of the population.

Config	Average Fitness
Roulette 20	20.57
Roulette 60	12.56
Roulette 80	15.22
Roulette 120	10.95
Cut 20	29.05
Cut 60	28.02
Cut 80	62.45
Cut 120	54.19
Tournament 20	21.16
Tournament 60	15.6
Tournament 80	15.22
Tournament 120	17.06

Table 4: Fitness by settings

The average fitness represents in a proportional way the average performance of the individuals in the round. By analyzing each selection method separately, it is possible to see that even the worst results of the Cut method are still better than the ones best adapted using Tournament and Roulette, as shown in Figure 4.

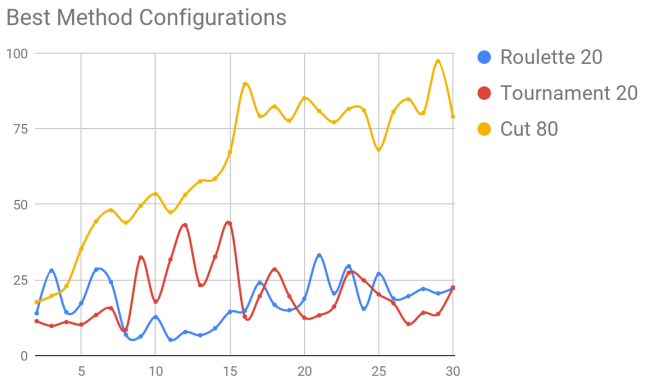

Figure 4: Comparison between selection methods

This behavior is due to the fact that the selection by the Cut is the most demanding and will always pass only the most adapted individuals to the next generations. In addition, it is well known that the Roulette and Tournament methods tend to take a larger number of generations to find optimal competitors. In contrast, they tend to fall into local optima with a lower frequency than the Cut method.

Based on these preliminary results and considering that games must obtain results in a short space of time, new tests were done. They, in turn, are restricted to experiments of settings with the Cut method, whose configurations can be verified in Table 5.

Population	Selected
20	5
20	10
20	15
60	15
60	20
60	30
60	45
80	10
80	20
80	30

Population	Selected
80	40
80	50
80	60
120	10
120	20
120	30
120	40
120	60
120	80

Table 5: Different settings for the cut selection method

After running these new experiments, some results of the algorithm were selected from settings that stood out. These results can be seen in Figure 5.

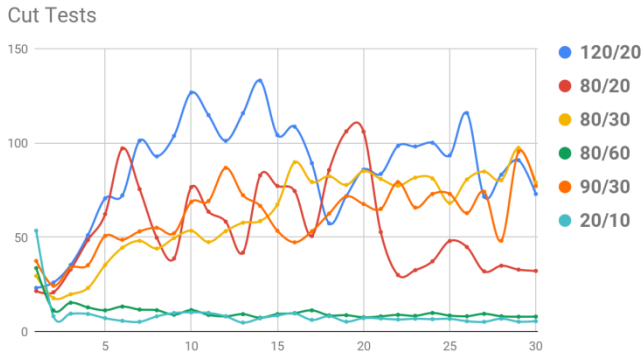

Figure 5: Best configurations for the cut selection method

The two configurations with the worst adaptation results were: one with a population of 80 individuals of whom, 60 were selected as survivors, and other, with 20 individuals from which, 10 were selected.

The configurations with 80 individuals were the most varied in general, spreading all along the search space, according to the number of individuals selected at each generation. The cut to the best 20 individuals, for example, even though it does not show a good final adaptation, shows excellent results in the generations between 15 and 20.

The configurations with 80 individuals cut to the 30 best individuals and 120 individuals with the cut positioned among the first 20 have the best results obtained in the experiments. Both have good results, but different characteristics. While the Cut 80/30 method has almost linear performance and few significant changes in fitness averages, the Cut 120/20 has had several ups and downs over the generations.

The behavior of the chart may have been caused by the small number of individuals selected, which tends to result in loss of genetic diversity. With this loss of diversity, a premature convergence of individuals occurs, which leads to an increase in the mutation rate of the algorithm. In this way, the next generations become more random and fall again.

When it comes to gaming applications, however, a configuration is needed that will quickly achieve satisfactory values. Therefore, it was decided to use in the game the configuration with 120 individuals of which the best 20 are selected. In this way, the game tends to be more challenging for the player, even for the first runs.

5 Experiments

The experiments involved 18 players - with previous experience in this kind of games - whom after taking the test of the game, answered a questionnaire about it.

These participants played until their main character died. In relation to the number of hordes each one survived (which can be seen in Figure 6), the behavior approaches a normality curve, with the average number of rounds surviving about 12.

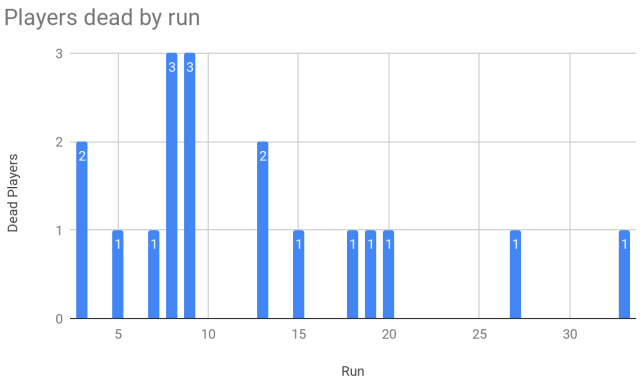

Figure 6: Players dead by run

As the hordes are composed of only 20 individuals, the generations go through every 6 hordes. This means that most players have been defeated by members of the first two generations of the algorithm. And even the player who survived the highest number of rounds, was defeated by individuals of the sixth generation of the algorithm. Still, on the performance of the algorithm, Figure 7 demonstrates the test time (also playing time) of each participant.

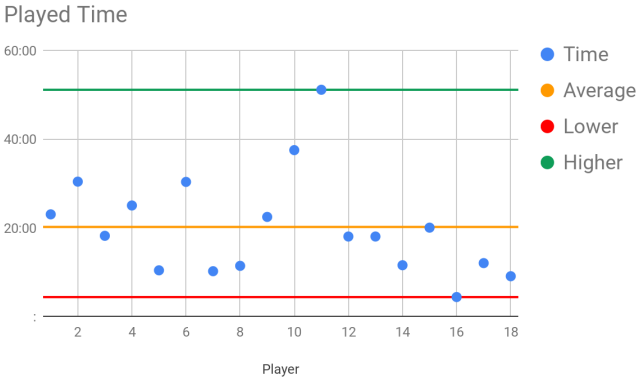

Figure 7: Played time

In this case scenario, it is noticed that on average the players remained alive for 20 minutes, which leads to an average time of one and a half minutes per horde (considering the average of 12 hordes). One of the players survived the game for only four and a half minutes, ultimately dying in the second round. Firstly, it was assumed that this situation was due to the fact that the initial population (which is generated randomly) had been very fit during this execution.

By conducting a more detailed analysis of the execution however, it was possible to discover that unlike the initial hypothesis, the first population generated in this execution was slightly below the initial fitness average. The player had died prematurely in the second round for reasons of mechanics of the game. Not having understood the operation of the weapon grenade, he ended up committing suicide.

On the evolution of the genes, a standardized analysis was made where the algorithm fitness is compared with the characteristics of the individuals. Figure 8 demonstrates this comparison for one of the tests performed with players.

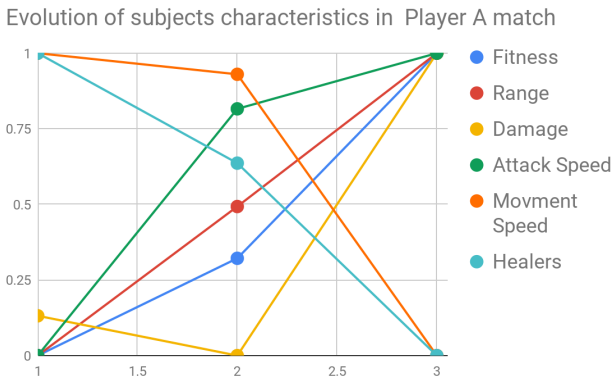

Figure 8: Evolution of individuals characteristics in Player A match

It is possible to perceive that some characteristics behave in a way directly proportional to the fitness, while others, in an inverse way. Thus, as the algorithm evolves, the parameters define the enemy attack's damage, distance and speed. In contrast, healers and enemy's movement speed decreases. The chart presents an interesting behavior because it intuitively should be increasing. This expected behavior can be seen in Figure 9.

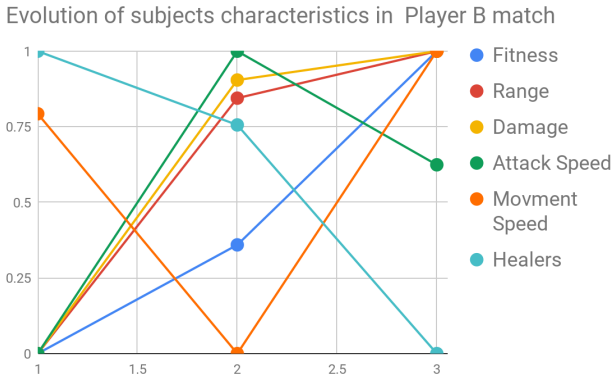

Figure 9: Evolution of individuals characteristics in player B match

As in Figure 8, the characteristics of the individuals in Figure 9 are some direct and others inversely proportional to the fitness level calculated by the genetic algorithm. In this experiment, however, it is possible to perceive a greater convergence of the characteristics in the second generation. With the exception of the Speed gene, all of them had a considerable increase in the second generation.

This made player B grow much faster than A, as it can be seen in Figure 10.

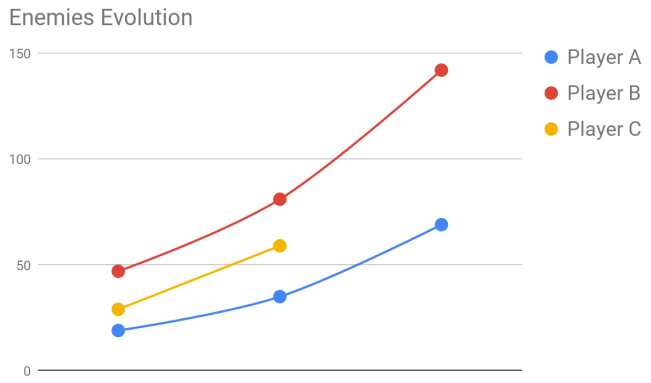

Figure 10: Evolution enemies' performance in different executions

As predicted in the settings tests, a large population with an elite selection demonstrated accelerated growth in executions, which allowed players have an overall growing game difficulty experience, which makes it more interesting to players [7].

When asked about the difficulty rate in defeating the enemies (Figure 11), most players replied the difficulty of the game as normal.

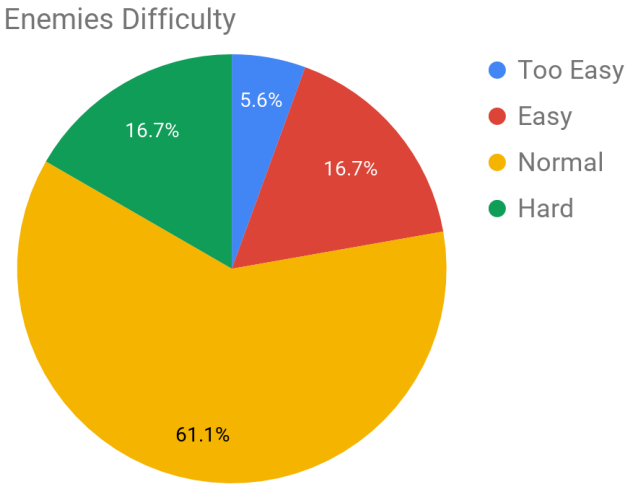

Figure 11: Difficulty in defeating enemies according to players

No player considered the game as Very Hard, and only one considered it Very Easy. This tendency to Normal difficulty indicates a certain flow in the gameplay, which in turn meets the expected result in relation to the gaming experience. The goal is to make the game more challenging, but not frustrating.

In relation to the change in difficulty of the game, Figure 12 shows the impressions of the players.

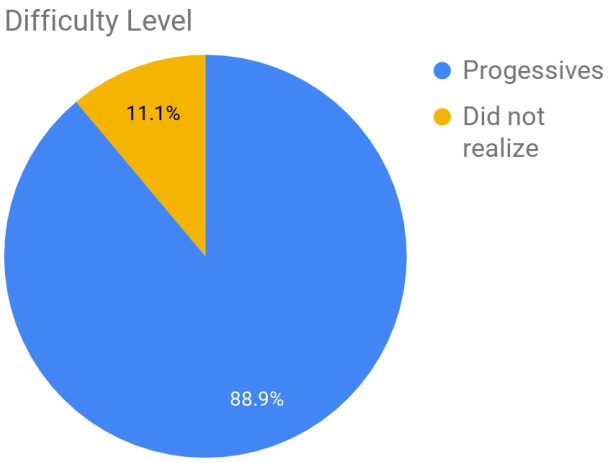

Figure 12: Difficulty evolution

Even with the algorithm evolving few generations for most players, as the adaptation is accelerated, players might have noticed a progressive increase in the difficulty level of the game. Although no player has defined the changes in difficulty of the game as abrupt, some players did not realize this change. It is believed that these are the ones who have not passed the first generation of enemies and thus have fought only against randomly generated enemies.

6 Conclusions

The Genetic Algorithm implemented in the Survival Shooter game is a good alternative for adjustment of difficulty in games of this kind, as it is able to offer a dynamic and challenging environment to the players. Although, other alternatives like NEAT algorithms (with Genetic Algorithms and Neural Networks), should perform better.

The players were able to notice the adaptability of the game difficulty. Unfortunately, the research did not count on the participation of inexperienced players, and it is not possible to affirm the adaptability of the algorithm to this public.

REFERENCES

[1] Ben Coppin. 2015. *Inteligência artificial*. Grupo Gen-LTC.
[2] Charles Darwin. 1859. *On the Origin of Species*. Routledge.
[3] Andries P. Engelbrecht. 2007. *Computational Intelligence: An Introduction*. John Wiley & Sons.
[4] John H. Holland. 1992. Adaptation in natural and artificial systems. 1975. *Ann Arbor, MI: University of Michigan Press and* (1992).
[5] Ricardo Linden. 2008. *Algoritmos Genéticos (2a edição)*. Brasport.
[6] George F. Luger. 2004. *Inteligência Artificial-: Estruturas e estratégias para a solução de problemas complexos*. Bookman.
[7] Jeannie Novak. 2010. Desenvolvimento de games. *São Paulo: Cengage Learning* (2010), 354–355.
[8] Raymond Redheffer. 1948. A Machine for Playing the Game Nim. *Am. Math. Mon.* 55, 6 (June 1948), 343–349.