Artificial Intelligence for Game Developers

By Michal Piechowiak

Student No. 17062683

# Game Description

To complete the assignment, I needed to create a simple game. To do so I have used a game engine called Unity. In the game we have a player object which is a spaceship, which can move only horizontally, and we also have an asteroid which is falling vertically. The whole point of the game is for the player to move and try to dodge the asteroids (if player doesn’t dodge, player loses), what was inspired by a mini-game of a game called “Warblade” made by Edgar M. Vigdal (my version of the mini-game is much more primitive).

First, players always spawn in the bottom middle. The asteroids are being generated randomly, but every time the game is played the asteroids are spawned in the same order, just like when using seed which is not time depended (something like 12345 as a seed will always give same output every time program is being executed). At last important thing to mention is that the game cannot be completed as it is a score-based game so there is no target to reach other than beating the highest score.

I choose to create such a game to test the AI I was implementing because I tried to play the “Warblade” mini-game at one point and I found it difficult to complete it. Therefore, I wanted to see if an AI will be capable of completing such a game or in my game’s case how high it can get

# AI Description

AI that I have chose for my game is a Genetic Algorithm which is inspired by the theory of natural evolution created by Charles Darwin. To achieve the goal, we use a process of natural selection in which the fittest members survive and reproduce the next generation.

So, in my case I am creating around 500 game objects which are spaceships. Therefore, each spaceship has its own “DNA” and “genes”, where “DNA” is a class that stores the fitness of an object and “genes” stores the direction in which game object can move. As mentioned before spaceship can only move horizontally that is why I have chosen the “genes” to be created as a list of floats.

To achieve success there need to be a way to somehow control the population that is being created each generation. First population is fully random as every single gene is picked from a random range between -1 and 1. But for every further generation I have a bit of a control as I was able to pick the fittest objects, which were determined by the time they manage to survive, and keep them to generate next generation (I kept 30% of the fittest member of population), some of the survivors where fully copied to the next generation (around 10% of a whole population) so the progress is not lost. So, the way that I have created further generation is that I created a for loop, which allowed me to send each survivor (equal amount of time) as a parent and one other survivor as the 2nd parent (which is picked randomly) to the “DNA” class until the whole population was re-generated, as soon as the “DNA” received both of the parents it started to generate a new “genes” which had 2 options mutate (5% of chance) or be crossed over. So, the way that mutation works is that every time we hit that 5% a totally random “gene” will be created, on the other hand I also mention cross over and all there is to it is that “DNA” with a 50:50 chance chooses which parent to chose to generate a “gene”.

# Conclusion

To conclude I want to say that AI that I have picked was a good choice to solve my problem as I was able to see improvements in the behaviour of the AI as shown in the table below. AS it can be seen I took 1st, 50th, and 100th generation, it clearly shows the improvement in the fitness of the fittest member.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Generation | Fitness | Score |
| Game log 1 | 1 | 420 | 12 |
|  | 50 | 920 | 24 |
|  | 100 | 1140 | 27 |
| Game log 2 | 1 | 410 | 12 |
|  | 50 | 1490 | 39 |
|  | 100 | 1720 | 43 |
| Game log 3 | 1 | 510 | 13 |
|  | 50 | 1340 | 33 |
|  | 100 | 1730 | 47 |

I am satisfied with my choice of an AI, mainly because even if the AI starts really poorly if we give it enough time it will come up with the solution.

# References

*Simple genetic algorithm tutorial in Unity3D(C#) part 1* by *UpGames* - <https://www.youtube.com/watch?v=1oXr16Tdfvo&pbjreload=10>

*Simple genetic algorithm tutorial in Unity3D(C#) part 2* by *UpGames* –

<https://www.youtube.com/watch?v=qWu3HkFgqfw>