Boxing

Artificial Intelligence

*AI-learning of agents to box*

Morgan Nyman

**Högskoleingenjör Datorpelsutveckling**

**2022**

Foreword

This report has been written during my education in Computer programming at Luleå University of Technology. The work has been done in 2022.

Skellefteå, May 2022

Morgan Nyman

Abstract

There are a wide range of approaches to use Artificial Intelligence (AI). One approach is to take inspiration from natural selection. Natural selection is about rewarding the most beneficial feature. Charles Darwin’s famous words, “*the Survival of the fittest”,* also applies when working with Artificial Intelligence because it handles variation in a similar way. A lot of different variables in the same field competes and the most beneficial survives. This method has created successful adaptation for all living organisms and is continuing to do so.

The method used in the project “Boxing AI” is inspired by the following report, *Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports (Won, Deepak, & Hodgins, 2021)*.The purpose of the project is to recreate and develop the AI further. The Boxing AI is programmed to select the most successful boxer among boxers created in a single batch. A batch is a variation of 40 boxing rookies (agents) with different boxing skills. Every time a new batch is created 20 boxing fights takes place in consecutive order. The boxing match lasts for 20 seconds, and the rookies are evaluated with scores. The score is rewarded in four categories. The four categories are, stance, balance, punching power and position. A higher score is considered a better boxer. When the next batch is created 20 rookies are eliminated and 20 rookies are included in the following batch. This process is repeated indefinitely.

Boxing AI is an ongoing process with no finish. If there are enough variations among the boxing rookies, they keep improving. The Boxing AI is a program with no ending which is like natural selection. When you add time, variation and competition improvement occurs.

This project is a about creating boxers out of boxing rookies in a virtual world and it has been a real challenge both in this project as well as in previous article (Won, Deepak, & Hodgins, 2021)

Innehåll

Foreword

Abstract

[1. Introduction 1](#_Toc101195862)

[1.1 Background 1](#_Toc101195863)

[1.2 Purpose 1](#_Toc101195864)

[1.3 Goal 1](#_Toc101195865)

[1.4 Delimitation 1](#_Toc101195866)

[1.5 Method 1](#_Toc101195867)

[2. Theory 1](#_Toc101195868)

[2.1 Previous work 1](#_Toc101195869)

[3. Empirical 3](#_Toc101195870)

[3.1 Project envirornment 3](#_Toc101195869)

[3.2 AI-learning 3](#_Toc101195869)

[4. Results 5](#_Toc101195871)

[5. Discussion and analysis 9](#_Toc101195872)

[6. References 10](#_Toc101195873)

# Introduction

## 1.1 Background

This report is a part of the program “Computer programming” at Luleå University of Technology. The assignment is to find and recreate a previous project and develop it further. This report is inspired by the work “*Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports”* (Won, Deepak, & Hodgins, 2021).

## 1.2 Purpose

The purpose with this project is to create Artificial intelligence (AI) in a similar way as in the report “*Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports”* and if possible, try to improve the AI even further.

## 1.3 Goal

In this project the goal is to create agents, and to apply AI and natural selection to help improve agents (rookie boxers) to become great boxers.

## 1.4 Delimitation

The limitation in this project was to only use a natural selection mechanism. It was possible to use backward propagation and other methods, but it was not used in this report.

When using the method *natural selection* in this project one delimitation is that the rate of improvement is due to random selection. From every batch it is possible for flawedvariations to live for a period before competition eliminates these variations. If natural selection applies there will always be flawed variations present.

## 1.5 Method

The project environment used is Unreal Engine. The environment consists of a physics Engine and a rasterizer in one application. This application combined with programming AI enables rookie boxers to improve.

The program is created with C++ as programming language. The editor in this project is “Visual Studio Code” by Microsoft Corporation.

# Theory

## 2.1 Previous work

In the paper *Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports* *(Won, Deepak, & Hodgins, 2021)*, two rookie boxers duel in both boxing and fencing. The agents (rookie boxers) learn to maintain balance and upright standing. The agents also build up power in their attacks to deliver as much physical damage to the opponent as possible and win the match. The fencer will try to hit the opponent's vest and force the opponent of the arena to secure a win.

The procedure of learning boxing has similarities to learning fencing in their project. The AI in their project is using a motion capture of real-life boxers that lasts for 90 seconds. Agents apply forces and torques to animate the duelists and invoke each duelist’s preferred tactic.

The agents will learn to recover their stance after taking a jab as well as use the ropes outlining the ring to assert speed.

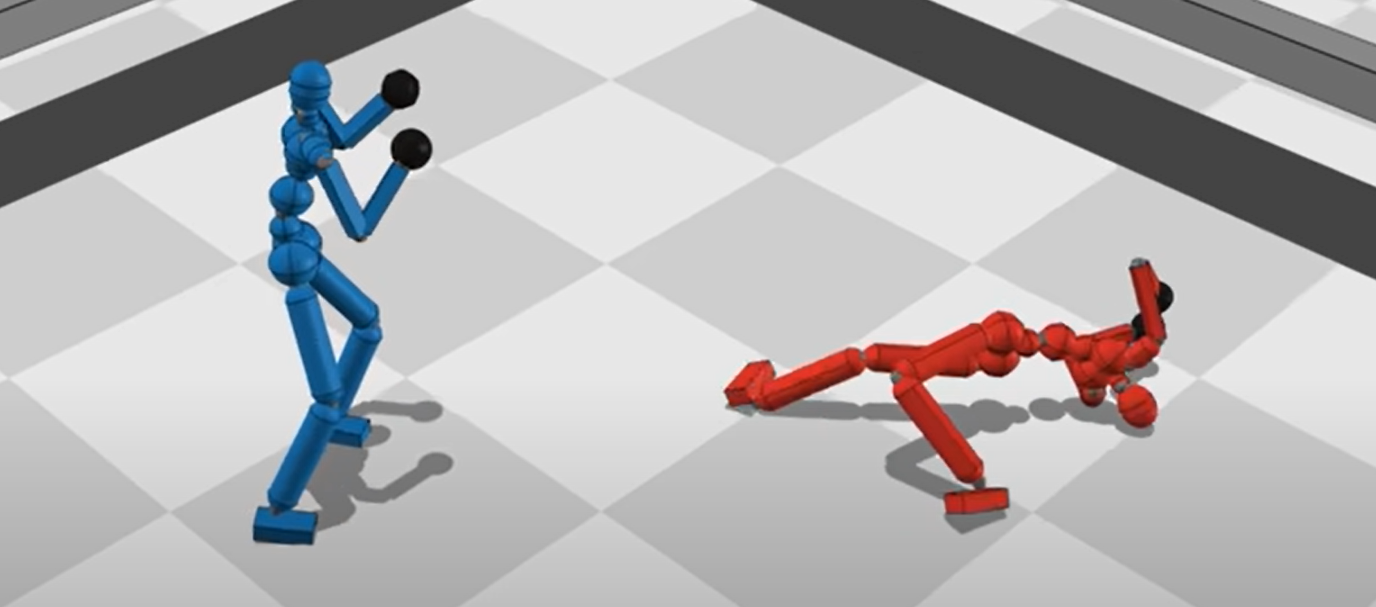
Different methods are used and after about 130 000 000 training steps their agents could maintain balance. At a whopping 200 000 000 training steps, the agents could strike the opponent with force (Won, Deepak, & Hodgins, 2021)   
  
The research yielded that the player’s movements are very slow compared to the real-life boxers. The greatest driving factor for the making of this project is to pave a path for follow-up projects to improve bipedal competitive sports with the help of AI.

En bild som visar text, bord, enhet

Automatiskt genererad beskrivning

*Fig 1: Characters performing two-player competitive sports such as boxing (left) and fencing (right) using learned control strategies* (Won, Deepak, & Hodgins, 2021)*, page 1*

In the project Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports (Won, Deepak, & Hodgins, 2021), the authors describe that in the first stage of AI learning the agents learn to stand upright for as long as possible. Eventually they fall (fig 2). Many tests later they learn how to stand without falling and then finally they start to learn boxing (Won, Deepak, & Hodgins, 2021).



*Fig 2: The player falls after some time of standing upright* (Zsolnai-Fehér, 2021) *time: 1m33s*

# Empirical

3.1 Project environment

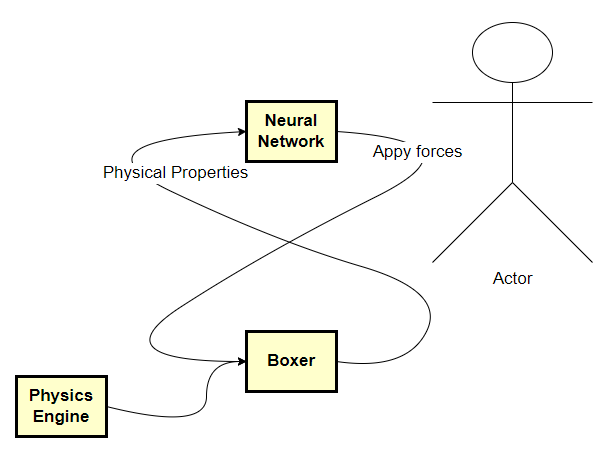
In this project Unreal Engine was used. It is a physics engine and rasterizer in one application. It was chosen because it made it possible to set up the environment neatly which allowed for more time to spend on producing the AI and boxing scene.

3.2 AI-learning

A Neural Network is a mathematical graph, which is used to simulate artificial intelligence. Between every layer of a Neural Network, there is usually a weighted connection between each node on both layers. A node is a meeting point for weighted connections.

It was crucial in this project to have the right number of hidden layers. Too many layers risked overfitting, because the network was extremely prepared for a few cases and unprepared for other cases. Having too few hidden layers made it very difficult to get any progress done.

The methodology for natural selection in this project was originated from an episode of Evan’s YouTube-channel (Evan, 2022), named “How AIs learn part 2 || coded example” (Evan 2. , 2022) and from “Caryh” YouTube-channel (Evolution Simulator (Part 1/4), 2015).



*Fig 4: Loop of each tick*

Inside the physics engine Unreal Engine, there is an event called “Tick Event”. After certain duration, the code that the tick event calls are executed. Every actor/agent has one tick event that belongs to them (fig 4). The loop is executed every tick if that actor is spawned into the current scene. Tick event is described by Unreal Engine on their website “Actor Ticking” in “Unreal Engine 4.26 Documentation”:

"Ticking" refers to running a piece of code or Blueprint script on an actor or component at regular intervals, usually once per frame. Understanding the order in which your game's actors and components tick relative to each other and other per-frame tasks performed by the engine is essential to avoid off-by-one frame issues, as well as to ensure consistency in the way your game runs. Actors and components can be set up to tick each frame, at set minimum time intervals, or not at all. In addition, they can be grouped together at different phases in the engine's per-frame update loop, and can be individually instructed to wait for specific ticks to complete before beginning. (Unreal Engine, 2022)

This project capped framerate to a maximum of 15 frames per second. Giving approximately 66 milliseconds to complete the computations of the neural network and other miscellaneous computations, which is the lowest framerate limit allowed by Unreal Engine.

In each tick the gravitational force pulled the agents down. To some extent countered by the normal force with collision to the ground. The body limps are physically restrained, to look realistic. There were also multiple forces generated by the neural networks output layer, which acted upon the agent - one force vector for each body part. The physics behind Unreal Engine’s physics engine is non-deterministic. That means that it is not possible to calculate the result beforehand. An effect of that is that the resulting positions and velocities of each frame may vary slightly. The fact that the physics engine is non-deterministic can also affect the final score.

# Results

In this project the purpose was to use AI to teach agents how to box. The “Boxing AI” project was programmed to select the most successful boxer among boxers created in a single batch. A batch is a variation of 40 boxing rookies (agents) with different boxing skills. Every time a new batch was created 20 boxing fights took place in consecutive order. The boxing matches lasted for 20 seconds, and the rookies were evaluated with scores. The score was in four categories: stance, balance, punching power and position. An agent with higher score was considered a better boxer. When the next batch was created, 20 rookies were eliminated, and 20 rookies were included in the following batch. This process was repeated indefinitely.

“Boxing AI” can be an ongoing process with no finish. If there are enough variations among the boxing rookies, they keep improving. The “Boxing AI” is a program with no ending which is like natural selection. When you add time, variation and competition improvement occurs.

The details of the working process describes as follows:

Setting up the scene:

1. Crafting the agents from cube and sphere prefabs.
2. Building boxing ring from cube prefabs.

AI-training:

1. The positions and velocity of each body limb of the boxer were stored, to be used as input into the Neural Network.
2. There was a random set of weights generated for each of the hidden layers (stored in files with .ih1, .h1h2 and .h2o as file extensions). The inputs were multiplied by a unique weight for each connection between adjacent layers. Until the last output layer, where the resulting values were returned as forces to the limbs of the agents.
3. This happened in the tick event that was run every 16 millisecond and run continuously alongside the boxing match.
4. After 20 seconds elapsed, the match was terminated, and both agents were given a score depending on their performance. (stored in a .score file)
5. The score was then used as a probability of the rookie boxer being killed in the upcoming natural selection. Either they were too unfit to fight and would die, or they were very fit and would play on.
6. A new round was initiated, and a new match begun with agents who had a new arrangement of weights in their neural networks.
7. The same procedure repeated from point 1-6, 20 times.
8. When 20 matches had been played, all the agents of that batch were gathered to be picked for procreation or termination. One batch consisted of 20 matches.
9. The terminated agents left an open position. In the next batch, this open position was filled with a new agent that was cloned from a surviving agent considered to be fit for fighting.
10. All agents in a batch (except for the most fit one) got mutated a small amount of their weights before they went into the next batch. The agents that were terminated did not proceed to the next batch.

In this project, several batches have been executed. Even after a few batches of matches, there was an increase in the median score among the agents. However, the scores were inconsistent, due to the fact that Unreal Engine (the physics engine) is non-deterministic.

In the project the agent had a force applied to them through neural networks, inside the physics engine with randomized weights for each agent. The force pushed a body limp of the agent so that the agent moved. Next tick, the agent was in a new place with a new velocity. These new values were inputs to the neural network in the next tick. This repeated during the whole match. After the match the agents received a score, which was important for the evolution of agents. The favored weights of each batch were more likely to be copied over to the next round, but not guaranteed.

There was quite a bit of data being stored away after a batch. The data being stored was of type float which was a 4 B (bytes) floating decimal number. There were 22 743 floats stored by each agent with a total of 40 agents in one batch that would account for 3,6 MB (3,638880 Megabytes)

The following figures shows the learning process of AI in this project:

En bild som visar golv, inomhus, vit, schackpjäs

Automatiskt genererad beskrivning

*Fig 5: Both agents start the match standing opposite each other, diagonally across the ring*

En bild som visar golv, inomhus, vit, schackpjäs

Automatiskt genererad beskrivning

*Fig 6: Both agents begin to descend due to the force of gravity outweighing the cumulative upward force of the neural network*

En bild som visar golv, inomhus, vit, ljus

Automatiskt genererad beskrivning

*Fig 7: The entire weight of the agent has hit the ground and is continuously penalizing the agent for not leaving the floor*

En bild som visar golv, inomhus

Automatiskt genererad beskrivning

*Fig 8: One of the agents gathers enough speed to roll of the platform*

The first step of the project was to create agents. Next step was to teach them how to move by AI-learning and natural selection. The first two steps were successful. The agents were created, and they could move.

It has been demonstrated that the agents may move due to the inputs of the neural network. There is a correlation between the agent’s movement and the movement of the opponent. Calculations are being made from the information given about the position and velocity of themselves and their opponent. It is as if they make an intelligent decision.

However, the agents could not stand successfully, they fell to the ground; some escaped the boxing ring. Also, by the end of this project, they still could not box.

# Discussion and analysis

In this project the purpose was to teach boxing with AI. The first step was to create agents. Next step was to teach them how to move by AI-learning and natural selection.

In this project, several batches have been executed. Even after a few batches of matches, there was an increase in the median score amongst the agents. However, the scores were inconsistent.

The first two steps were successful. The agents were created, and they could move. However, the agents could not stand successfully, they fell to the ground; some escaped the boxing ring. Also, by the finish of this project, they still could not box.

A similar problem occurred in the AI learning of “Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports” (Won, Deepak, & Hodgins, 2021): The agents learnt how to stand, but then they fell. Finally, after executing a great number of AI-learning batches, the authors managed to teach the agents how to really box by natural selection. Different methods were used and after about 130 000 000 training steps their agents could maintain balance. At a whopping 200 000 000 training steps, the agents could strike the opponent with force (Won, Deepak, & Hodgins, 2021).

In this project the implementation has about 100 training steps. In the short time span for this project there was not enough time to do as many trials as needed to teach the agents to really box.

Given plenty of time there is a possibility, in theory, to evolve agents that can compete on a professional level. In practice, this time frame was infeasible, the work of the Facebook/Meta research team described in “Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports” (Won, Deepak, & Hodgins, 2021) was an incredible achievement. And it took them months of around the clock training the AI to reach their goal. In this short time for project, it was realistic to accomplish to create actors who can slightly move.

One approach to continue this project could be to train the agents through a method of backpropagation. To divide the learning into smaller sections where the agents first learn to balance on their feet, and only once this skill is mastered, the boxer will start learning how to walk When the boxer must approach something, you could begin to call it a walk, first at this point does the actual full boxing experience training begin. The realism of the project could also be improved, if there was a limit set to how strong the forces acting on a limb can be so that it remains realistic.

# References

Evan. (2022, Februari 25). *youtube*. Retrieved from Code Bullet: https://www.youtube.com/c/CodeBullet

Evan, 2. (2022, februari 25). *youtube*. Retrieved from How AIs learn part 2 || Coded example: https://www.youtube.com/watch?v=BOZfhUcNiqk

*Evolution Simulator (Part 1/4)*. (2015, Juni 24). Retrieved from YouTube: https://www.youtube.com/watch?v=GOFws\_hhZs8

Unreal Engine. (2022, April 18). *Actor Ticking*. Retrieved from Unreal Engine 4.26 Documentation: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Actors/Ticking/

Won, J., Deepak, G., & Hodgins, J. (2021). Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports. *ACM Trans. Graph*, Vol. 40, No. 4, Aug 2021.

Zsolnai-Fehér, K. (2021, Aug 29). *YouTube*. Retrieved from This AI Learned Boxing…With Serious Knockout Power! 🥊: https://youtu.be/SsJ\_AusntiU?t=93