Final Report

# Introduction

## Description of Problem

A.I. research has expanded over the last decade into areas not previously possible. This is largely due to increased processing speeds and increased data storage possibilities. With this expansion came new techniques that are used to train and test artificial models. The area that my project will be focused on is genetic algorithms. Genetic Algorithms are a form of reinforcement learning that trains a model over repeated testing in an environment. I will be creating a testing environment that the user/player can use to experiment with and try out different ideas on how best to train an agent given different environments.

The game

## Beneficiaries

I identify three main beneficiaries for this project that will find a use for it. The first and simplest beneficiary is someone looking to have a bit of fun and try out some ideas in a field that may be new to them. I aim to make the project easy to approach and understand so that no training is required to start training A.I. models. The second beneficiaries are people who hope to experiment with new models and techniques without having to create a environment for them. Some environments already exist to benchmark new models, such as the pole balancing task, however I hope to allow for a greater range of experimentation. The third and final beneficiary is me. Through creating this project, I will be able to learn and experiment with techniques I have been looking at but have not used. I expect to be able to apply these techniques to any future work I undertake.

## Assumptions

I make three main assumptions when starting this project. Firstly, I assume I will be able to replicate the NEAT algorithm and adapt it to work within Unity’s game loop. If I cannot I assume I will be able to find a library that accomplishes this for me. My second assumption is that once the NEAT algorithm is implemented, I will be able to adapt it to the rtNEAT algorithm. This is because all of the network merging logic is the same. The final assumption is that I am prioritizing the workings of the game and not the aesthetics, but if required I will find game art and sounds online.

## Project Objections

I have split my project into three iterations, each of which will be explained below in more detail.

The first iteration will be focused on getting the bare bones of the project working. This includes setting up source control and implementing the NEAT algorithm. After this iteration the project should be able to run experiments and it will be possible for a layman to train a model from scratch.

The second iteration will be involve refactoring any parts of the game that reference the NEAT algorithm directly and allow it to work from defined interfaces. Using these interfaces, I will then create the rtNEAT algorithm that will allow for training in real time. The final part of this iteration will be creating a number of scenes the user can load and play around with.

The third iteration will be focused on adding basic gameplay and UI elements. This will be where the final project takes shape, and the game becomes more fun and useful to use. I have chosen to priorities this last as it will only be expanding on iteration one, and the project will still be useable before this is completed.

### Iteration One – Neat Algorithm working

1. **Basic Unity project setup:** This will involve creating a new unity project and adding some basic objects to the scene.
2. **Setting up Source Control:** I will be using GitHub and Git for source control of this project.
3. **Creating an Agent that will interact with the scene:** The game will require an agent prefab that the algorithms can use to interact with the environment. This will include:
   1. **Basic movement**
   2. **Collision Detection**
4. **Implementing the NEAT algorithm into the environment:** Either through direction implementation or using a library, I will add the NEAT algorithm into the environment and allow it to interact with the agents.
5. **Creating a very basic UI:** This UI should allow the player to save, load, create and destroy new experiments. It will also display basic information such as current best fitness and current generation.
6. **Unit tests created:** I will be creating unit tests to ensure that this iteration is working as expected.

### Iteration Two – Refactor AI code sections to use interfaces and implement rtNEAT.

1. **Abstracting NEAT algorithm from interacting directly with agents:** In the first iteration the NEAT algorithm will be directly connected to the agent’s and will be controlling them this way. For this step I will need to create an API that a generic algorithm can use to control the agents.
2. **Create the rtNEAT algorithm, using what I made in iteration one:** This iteration will require two main parts. Firstly I want to abstract away any direct references to the NEAT algorithm that were introduced in the first iteration. Then I want to implement the rtNEAT algorithm, using a lot of the backend implemented in the first iteration.

### Iteration Three – Basic gameplay and UI elements introduced

1. **Create a basic UI to show the user simple information.**
2. **Allow the user to create new experiments and choose the scene they want to use.**
3. **Allow the user to choose which algorithm they want to experiment with.**

# Output Summary

* 1. Unity Project
  2. Test Cases
  3. User Guide

# Literature Review

(COPY IN WHEN READY)

# Method

## Software Engineering Methodology

* Using Agile and iterative methodologys. I have worked within Units workflow of creating gameobjects that are attached to the scene that can then call further scripts to perform more complicated logic.
* I have strived to use interfaces as much as possible where any algorithm is concerned. This gives a much greater freedom for further expansion.
* I have used Git and Github for source control of the project

## Tools

* Unit
* Github
* Jetbrains Rider
* Visual Paradigm
* UnitySharpNEAT

## Iteration One

**This first build was to setup a basic Unit project with source control and implement the NEAT algorithm.**

* + 1. Setting up the Unity project and a Basic Scene
    2. Implementing the NEAT algorithm into the Unity scene
    3. Allowing for very basic control of the scene and Experiment

## Iteration Two

**This build involved generalizing the codebase to work with an IAlgorithm and IAlgorithmController. Then it was to implement the rtNEAT algorithm**

* + 1. Creating interfaces that can be used where any reference to the NEAT algorithm was used.
    2. Updating the old code to use these interfaces
    3. Using the interfaces to implement the rtNEAT algorithm

## Iteration Three

## Testing

# Results

## Software Engineering Methodology

## Iteration One

## Iteration Two

## Iteration Three

## Testing

# Conclusions And Discussions

# Reference List

# Glossary

# Appendices