IMPOSSIBLE

CO600 Group Project

—

Technical Report

Team Members

Bradley Stevenson - bts21@kent.ac.uk
Edgar Spencer - el324@kent.ac.uk
Matas Kimtys - mk750@kent.ac.uk
Max Retter - mr585@kent.ac.uk

Abstract

This report will go into how we developed Impossible, impossible is a round based aim trainer style game where the targets (enemies) are controlled via a rudimentary AI which we call *behaviour profiles.* These behaviour profiles can then be developed via the use of a genetic algorithm (GA) to incrementally make each round slightly harder than the round before.

1 Introduction

With the growing number of games available to assist users to improve their skills within first person shooters (FPS) as a collective we noticed that these games can often become dull reliant on user memory and no unique progression leaving users forced to choose difficulty settings which makes the targets act in the same way each time played and isn't reflective of how targets act during real gameplay with real human players. We decided there was a need for creating an aim trainer style game in which the user would keep progression whilst the difficulty would increment naturally to ensure that each individual that played Impossible would gain a unique experience based on their own ability, not reliant on muscle memory and memorisation of the layout of the targets, but purely the player's response to unexpected movement developed by the GA. In order to achieve this our game is based around the understanding that a GA will make the game's difficulty naturally adapt based upon each individual's play style. Another benefit of this game is to be used in the education sector to get students excited about computer science and to show, in an intuitive way, how GA's work.

2 Aims (Scope)

2.1 Database

Aim, to be able to save individual progress, allow users to continue where they left off, leaderboard to make it competitive among users.

2.2 AI as targets

Create targets where the movement of the targets can be represented in such a way where that can be evolved using a genetic algorithm.

2.3 Keeping the users interested

Make a good visual presentation of our environment, make user friendly UI, evolving targets make it so the user is always challenged and encouraged to pass the previous performance limit.

3 Design (development decisions)

3.1 Target Movement

We decided to create targets that can move unpredictably along predetermined behaviour profiles, on creation of the targets on round 1 these profiles should be randomised within a given bound this is in order to make sure that the genetic algorithm can't just try and maximise the distance between two positions and also minimise the time between each position this should have the effect of finding the best behaviour not the fastest movement.

3.2 Visuals

We had decided to implement blank wall textures from the beginning of our project also using a sphere as our object as our main focus was to ensure that we could implement our core ideas of how we wished the game to function. At a later stage we decided that we needed to keep players interested so we had implemented

a lot more visual effects to make the game look better. Using 3D gun animations to show gunfire and adding effects to the spheres tracking their movements along with having them combust on being shot.

3.3 UI

To catch the users interest straight from the beginning, we decided to animate the background of our login menu which we derived and tweaked from the poster. To keep the user interested, a good visual presentation of the UI is important. Therefore the mentioned tweak includes a glitched screen tearing effect that appears in the background. This was achieved with Adobe Photoshop and a function that recursively calls itself at randomised choice of wallpapers that each have a set time for display. Main menu and highscore menu is just a gunshot shattered glass effect black screen background. For inputFields we used the default. Our own hand made custom font included, it is briefly utilised in the login menu to write down “Username” and “Password”.

3.4 Database

Database location –
SERVER=dragon.kent.ac.uk;
DATABASE=mk750;
Tables- PlayerCredentials, PlayerProfile, PlayerTargetTable, potentially more.

The idea for the PlayerCredentials table is meant to be used as an individual profile page firewall. It will be used to assemble the login button functionality. (Fig. 1)

```
CREATE TABLE PlayerCredentials (
  PlayerNo INTEGER,
  Username VARCHAR(16) NOT NULL,
  Password VARCHAR(30) NOT NULL,
  PRIMARY KEY (Username),
  UNIQUE (Username)
);
```

(Figure 1)

The idea for the PlayerProfile table is to record any useful player gameplay information over play time and potentially at some point utilize it in our genetic algorithm for target to be shot difficulty increase. (Fig. 2)

```
CREATE TABLE PlayerProfile (
    Username VARCHAR(16) NOT
NULL,
    TargetsDestroyed INTEGER,
    timesW INTEGER,
    timesA INTEGER,
    timesS INTEGER,
    timesD INTEGER,
    WeakestTarget INTEGER,
    StrongestTarget INTEGER,
    AverageTarget INTEGER,
    Accuracy INTEGER,
    AimingSpeed INTEGER,
    TimeSurvived INTEGER,
    TotalGamesPlayed INTEGER,
    FOREIGN KEY (Username)
  REFERENCES
  PlayerCredentials(Username)
  );
```

(Figure 2)

The idea for PlayerTargetTable is to record any parameters of any surviving targets that after a specific amount of time are still alive, therefore additionally using this data for further genetic algorithm tuning. (Fig. 3)

```
  CREATE TABLE
  PlayerTargetTable (
    Username VARCHAR(16) NOT
  NULL,
    TargetNo INTEGER,
    Generation INTEGER,
    Speed INTEGER,
    Acceleration INTEGER,
    Size INTEGER,
    Executions INTEGER,
    Weight INTEGER,
```

```
  Triggers INTEGER,
  CooldownTime INTEGER,
  Abilities INTEGER,
  LastAbility INTEGER,
  PositionX INTEGER,
  PositionY INTEGER,
  PositionZ INTEGER,
  TimeAlive INTEGER,
  Dead INTEGER,
  FOREIGN KEY (Username)
REFERENCES
PlayerCredentials(Username)
);
```
(Figure 3)

4 Implementation

4.1 Behaviour Profiles

Each target (enemy) has a specific movement behaviour attributed to it, which is unique. This is achieved by each target having an array of positions and an array of times of the same length, this is then populated with random positions and time within a given range, the game engine through the update loop adjusts the position of the target to the next position in the array in the corresponding time, for instance if a target was at point A (3,4,5), the next position was point B (2,4,3) and time A was 2.4 then it would take 2.4 seconds for that target to move from point A to point B.

4.2 Genetic Algorithm (GA)

4.2.1 Candidate Representation

(See behaviour profiles)

4.2.2 Selection

Typically, a selection method would require some form of fitness function to determine which candidates are better at fitting a given task. In the case of our game that is not necessary, this is because the user naturally selects which targets are bad by killing them. This then means that that target can no longer pass its genetic information to the next generation and only the ones that have survived will be used to populate the next round, increasing difficulty by doing so.

4.2.3 Crossover

Crossover is a way of mixing two candidates (targets) genetic information, in our case we use single point crossover which is where you take two candidates split them at a random point and take from candidate 1's first part and combine it with candidate 2's second half which give you an offspring to be put forward to the next stage. However, our candidates are represented by two separate arrays, so it was decided that, when a crossover does occur, it should choose an independent crossover point for the positions and the time to increase diversity. Although this could decouple the positions from their times. Another part of crossover is how often it should occur, as just mentioned. Typically, a relatively high rate is used, and we didn't change that, so we use a rate of 95%.

4.2.4 Mutation

Mutation is a method of introducing new genetic material into the population. The mutation method that we used was single point mutation. This is where you choose a random point and change the value of that point, in our case we choose a random value within the accepted bound. Our multi-array setup also showed some challenge for mutation too, what we decided was to not change positions and times together but to just choose one of them, this is because mutation is an inherently destructive process, destroying potentially good genetic and replacing it with a random piece. Due to mutation being an inherently destructive process

Typically a low mutation rate is used and we don't divert from that, so we used a rate of 20%.

4.3 Database

We have used dragon, which is the university's database server to store our database table and its data in. The database server is running MySQL and allows us to store player data, which then can be used for displaying player statistics and comparing the player to other players of the game. The user authentication is also done through the database; the user's username and password is stored inside the database.

4.3.1 Login Menu

We have considered that the most frequent approach saving individual logs or progress in nowadays UI design is to use a username and password method of logging into a personalised account. Therefore, we have implemented a username and password input fields, with a login button. The login button functionality is coded by actionListener, which upon clicked it retrieves a username and password inputFields child component data and then compares them to the database. We accomplish this by running through Boolean checks for both by iterating over an existing PlayerCredentials table, column username in the database. If a case sensitive username is found and matches to one on the database, it will then pull all other column data, I.e. PlayerNo, Password for that specific username row and compare once again to the inputField data. If it matches this comparison as well, it displays a confirmation of a match for a moment and proceeds to put the user into the main menu screen.
If either username and or password fields are empty, then the user will have the login button text replaced by "Cannot be empty".
If the username is entered with an unmatching password, then the login button text "login" is replaced by "Incorrect Password" and reverts back after a moment.
If a valid username and password are given, and no username is found on the database, then we proceed to create an account with the given credentials. We insert a new row into the PlayerCredentials table in the database at the next new player number increment and proceed to carry onto the Main menu screen.

<u>Current Flaws/potential risks:</u>

- Crashes clients if not connected to University of Kent Network VPN, need to create a timeout and a try catch.
- Usernames should not be Case sensitive.
- No cooldown on failed attempts, accounts are at a security risk of having their username & password cracked via external software.
- No last login information being noted. I.e IP of user etc.
- No email confirmation and delay for account creation, usernames could be exhausted by looping external software to create new accounts.

<u>Difficulties encountered:</u>

Sql injection into unity. Any SQL library could be handled with a simple NUGet package import, we would just encounter a compilation error due to Unity not being able to read SQL .dll files even though they are in the Library directory of the project. This has caused over 30 hours of research across the internet for potential solutions. One small comment in stackoverflow has suggested that we

paste the MySQLdata.dll file into the main project asset directory.

4.3.2 Main menu

Currently the main menu does not include any database functionality. But in the UI, the design idea for the main menu is that it consists of a user statistics, leaderboard, play, quit buttons that we would be able to display. We would add an action listener to each which would trigger database functions. For example clicking the leaderboard button would pull a whole leaderboard ranking system data table and display it to the rest of the users to review themselves or other people's performance/statistics. Clicking user statistics would pull data from the PlayerProfile table and be displayed to the user so they can review their own performance and statistics.

4.3.3 Leaderboard

Is a table to be yet made, meant for a player ranking system based on uploaded player statistics at the end of each game. This table could contain player username, data of all the columns of PlayerProfile table combined with data of all the columns from PlayerTargetTable the strongest AI parameters encountered by the player

4.3.4 User Statistics

Player profile table, upon clicking the button the PlayerProfile data is retrieved from the database and displayed in a statistics window for each individual user. This trigger for pulling data from the database would be based on a actionlistener button.

4.3.5 AI

PlayerTargetTable, this is meant for storing strongest encountered AI parameters and keeping it until it is replaced by an even better AI. Furthermore the idea is that if we are able to save and load AI parameters for each individual player, this can help the player base try each other's AI and see how difficult it is for them, with the chances of developing an even better AI.

4.4 Version Control

To manage the project between our group, we have used github where we each collaborated on the project. Github has allowed us to easily view changes that each team member is making. Github also acts as a safe way to store our project's source files as only the project maintainers are allowed to view and download the source code.

4.5 Quality Assurance

We have carried out quality assurance on all source code that we have written. We have done this by asking several test subjects to test our game; we have also made sure to test each others' changes as soon as one us has deployed a change to the Github repository. With every push to the Github repository, we have also reviewed each other's code to ensure the integrity of existing content within the game and to spot any potential issues that must be looked at.

4.6 Music and Sounds

We have made sure to use royalty-free music and sounds (from StoryBlocks - www.storyblocks.com) within the game to be in line with the national and international copyright laws. We have set out to create a horror-type atmosphere with an upbeat type of background music which we have achieved well with several tracks playing at once.

5 Testing

5.1 GA

We added some crucial testing to our

genetic algorithm by making sure the object would change colour based on level of mutation (red=0, blue=1,green=2,yellow=3) so we could determine that the mutation in the genetic algorithm was working as with all genetic algorithms it can take many instances to see a big difference, once this was implemented we also wanted to show that the crossover was also working so we changed the genetic algorithm to set mutations to 1 when crossover occurred this did indeed change too but at a much higher rate inline with what you would expect given the difference between mutation rate being 20% and crossover rate being 95%.

5.2 Database

The base testing methods used for testing the functionality were try catch and if else methods. Initially we would check retrieved data from unity client and database whether it is valid data i.e. whether pulling a string retrieves us a string. Then we cross referenced the database with unity, to ensure that information is received, compared and used correctly. The connection to the university network is tested with the .ping() method which returns true or false. If SQL query is incorrect, we receive a timeout exception from the database.

6 Background

Prior to us starting the project, some preliminary research and learning was needed to be done, as each group member had to contribute towards the game's source code and design.

6.1 Unity

Unity uses the C sharp programming language for all scripts, this means that each team member had to learn this programming language if they were not already experienced in it. The benefit of UNITY game development is the easy of development and the wide availability of various packages and artwork for this platform.

During the build of IMPOSSIBLE, we have imported several UNITY publicly-available packages, such as: particle effects pack, Universal RP (for post-processing), TextMeshPRO (for the GUI) and more. We have decided to add post processing using Universal RP because it matched our requirements of the game; we wanted the game to be eye-catching and colourful, and for the enemies to be identified quickly by the player, hence why we have added excessive post processing to highlight all elements within the game.

6.2 Reading and Research

Prior to us starting to work on this project, some background research and reading was required regarding UNITY, genetic algorithms and how the genetic algorithms work with crossovers, behaviour profiles, mutations and selection. This involved understanding how to manipulate genetic algorithms and working out as a group how to showcase this manipulation in a game-type format with the UNITY game development platform. Additionally, prior database knowledge was required as we had to connect our game to a database - to store all user data, etc.

7 Requirements

The primary requirement for this project was to demonstrate how the genetic algorithm works and how the enemy targets change based on their behaviour profile. We have achieved this by the targets starting off as a red colour, and every time their behaviour profile changes, they change in colour; they go from red, to blue, then to green and finally to yellow

which means they have reached their final mutation.

7.1 Usability
The project must incorporate a simple interface (graphical user interface), to allow the player to understand the game effectively and navigate through it easily. The heads up display should be simple, not too cluttered and provide the most important real time information to the player, accurately and in real time.

7.2 Correctness
The information displayed on the graphical user interface and the heads up display should be real time and accurate. The mutations should also be accurate and be updated every single round. The player must be notified on round change and should see clearly that mutation is happening and how the genetic algorithm is working.

7.3 Response Time
The response time should be near-instant to show the real time changes within the genetic algorithm. If there are any significant delays, it may provide an inaccurate representation of how the genetic algorithm works.

7.4 Efficiency
The source code and the game design must be efficient and effective. Since there will be a lot of enemies spawning and the genetic algorithm will be running behind the scenes, we have to ensure that the code is reusable and as simplified as possible, whilst still making it clear what the code does.

7.5 Code Reusability
The code should be reusable, meaning that if a change or a bug fix is ever required within the game, it would be easy to carry it out. The source code should be commented where possible and where the code does not make it immediately obvious what the purpose of it is.

7.6 Visual Effects
The game should have a considerable amount of visual effects to make the game enjoyable and a fun environment to learn how the genetic algorithms work. The game should preferably make use of post-processing, and the targets should be highlighted on screen and clearly visible at all times, to make the game fair.

8 Conclusion

Impossible has been a successful project overall showcasing the implementation of genetic algorithms learning how a game can increment difficulty based on the users skill. We made an effort to put our own take on what would essentially help the user improve their skills in shooting games to assist them in becoming more accurate with unpredictable moving targets. As a group we understood that the game should somewhat be interesting as with many aim trainer games available but fully predictable this takes away from the real gaming aspect of FPS games, it was decided that we would overhaul the visuals of the game in order to keep the user more interested. We made sure to add in copyright free music and even created a home screen and game over screen, with these additions added we intended for the game to feel more authentic as a fully functional game whilst serving a purpose of improving the users aiming within the FPS genres. The comparisons we found with other styles of these games was that they were focused mainly on learning the positions of the targets

whilst aiming to hit them as fast as possible whereas we wanted the user to actually learn and improve their overall gameplay and we feel as a group this has been achieved. A novel aspect of the game would be using genetic algorithms to control the movement of a non player character (NPC) as mentioned prior, candidate representation would be a challenge but we came up with the innovative behaviour profiles which can both be used for controlling targets' movement and are well suited to the genetic operators commonly used in genetic algorithms.

9 Future Work

For the future improvements, we have considered the following:

To contribute toward unpredictability of the aiming we would like to add target size variety. We would also like to improve the appearance of objects by adding sprites to the targets to make them more visually appealing over just smooth spheres. Improving user experience, we could have improved the look and feel of the UI by adding textures to the buttons and perhaps making them larger and more obvious. Our most delicate subject of our project is the GA of AI, it surely needs additional balancing as well as player and AI variables that should apply towards the results of GA. To achieve this we will improve our database tables which also includes adding a leaderboard that utilises PlayerStatistics table. This leaderboard we plan to display to the players as a separate menu in the player statistics page.
To improve our database security, all of its mentioned flaws are to be resolved, meaning we would create a custom database, adjust its functionality to make the data we receive properly secured by sanitising inputs and using parameterised queries. On multiple occasions we have discussed creating a level system that will let the user progress through different arena environments.

- A registration page, allowing the user to register appropriately.
- Options page, allowing the user to change the following: mouse sensitivity, sound effects, music.
- At the start of the game, there should be instructions on-screen; i.e. hit the targets, and how to move (controls).
- Make it an installable standalone client and publish.

10 Acknowledgements

We want to thank Dominique Chu for supervising our project, providing valuable advice every step of the way and ensuring that our project is on its way to completion and is in the right direction, and also ensuring that all of our team members stay focused and attentive; he has been a great help to our project and in ensuring the success of our team's goals. We would like to additionally thank Unity for providing such an easy to work with game development environment and we would like to thank the creators of the Unity packages that we have used,

as well as the creators of the music tracks that we have used throughout the game project.