George Andersen

QWOP in JavaScript

Computer Science Tripos – Part II

Robinson College

April 14, 2018

Chapter 1

Introduction

1.1 What is QWOP?

As described in the project proposal, my project is to recreate QWOP using JavaScript. QWOP is a browser video game created by Bennett Foddy. The goal of the player is to complete a 100m race without falling over. Figure 1.1 shows the screen on game start. Players control the athlete using only the Q, W, O, and P keys. Whilst the Q key is held

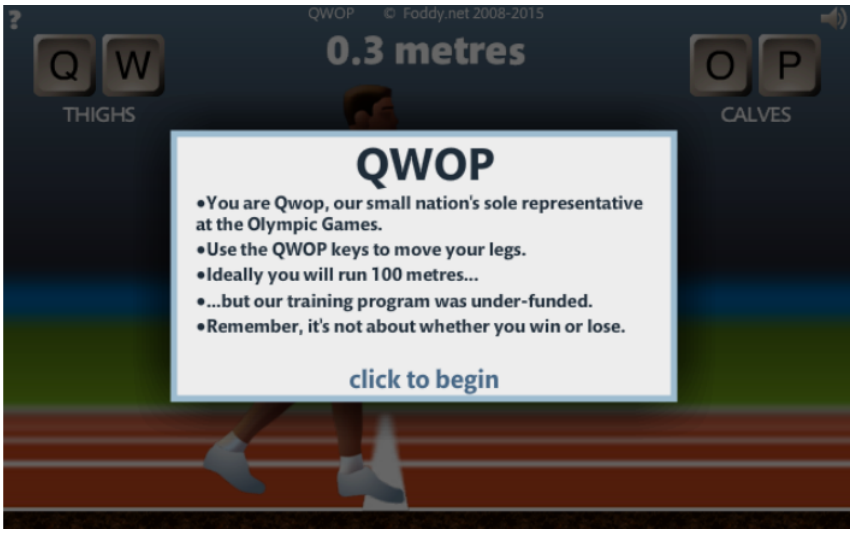

Figure 1.1: Starting screen of QWOP

down, the runner's right thigh is driven forward and his left thigh is driven backward. This moves the legs apart at the bottom of the torso in a scissor-like motion. The W key does the opposite of this, and holding it down will put the athlete in a similar stance but with his outstretched legs reversed. The O and P keys have a similar effect but instead power the movement of the calves. The O key applies a force that moves the left calf towards the back, and the right calf forwards until it is in line with the thigh. Again the P key is the reverse of this.

Once the player clicks on the screen they are free to press the keys to attempt to move forwards. This invariably leads to Figure 1.2 which shows the athlete fallen over. Below the athlete there is an X symbol. This shows where the athlete first touched the floor

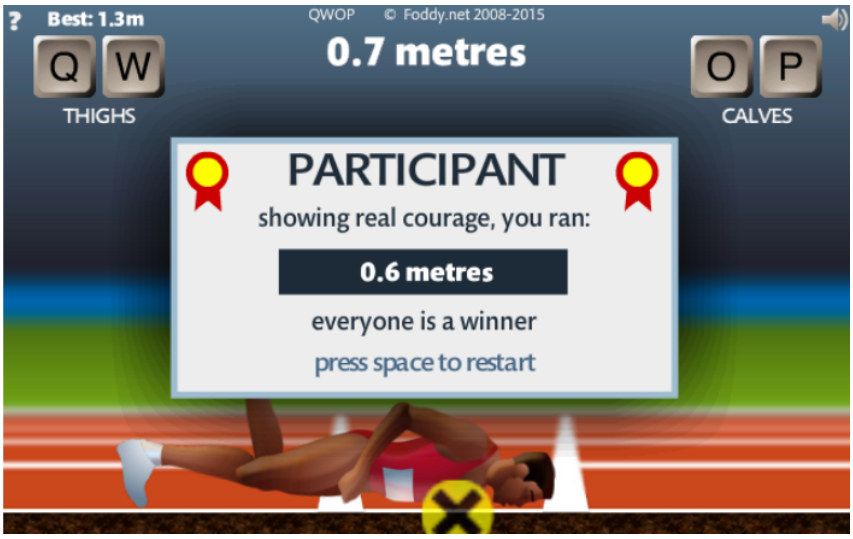

Figure 1.2: The athlete has fallen over so the player must restart.

with his arms or head. These section of his body are not allowed to touch the floor as doing so counts as falling over. After this the player can press space and start again. Most players attempt to keep the athlete upright and work out a key combination that will move the athlete in a motion as close to a normal gait as possible. Others however manoeuvre the runner into a stable position on the floor, and then press lots of keys in the hope that there is a net movement forward. Both these methods are entertaining and give some success. Though the objective of QWOP is simple, it has been notorious for being difficult to master ever since it went viral in 2010.

Chapter 2

Preparation

2.1 section 1

In this chapter I will discuss the preparation that happened before the implementation of the project began.

2.2 Mathematical model of the QWOP world

To create QWOP in a physics simulation it first helps to know how it is going to be modelled. So when preparing it was important to first design a mathematical model of the QWOP world that could later be implemented in a physics simulation.

2.2.1 Planar rigid body mechanics

The basis of this model is planar rigid body dynamics. As the floor that the athlete is standing on and the limbs of the athlete do not change size and do not need to deform, they can be modelled as rigid bodies. Save the floor, each of these rigid bodies represent one section of the athlete that has it's own position and rotation. For example the foot, upper leg and lower leg. Planar rigid body mechanics uses kinematics, and solves the equations of motion described by Newton's second law to determine the motion of the bodies in the system. Since this model can describe the acceleration, velocity and position of the individual rigid bodies over time, the development of the system as a whole under gravity can be simulated. With only basic planar rigid body dynamics, the limbs will just fall to the floor. So next I will describe some extensions to this model that will be made to emulate the behaviour of QWOP.

2.2.2 Joints

Adjacent sections are connected by joints. For example the foot and lower leg are connected at the heel, and the lower leg and upper leg are connected at the knee. This is a constraint that the joints force the members to follow. For each joint-body connection, the joint stays in the same position relative to the frame of reference of the body - joints keep a consistent offset from each connected member. For example, the heel is always at

the back of the foot, and the heel is always at the bottom of the lower leg. These two constraints work together in each joint to connect two limbs.

Another constraint the joints keep is that the angle the joints rotate to is limited. For example the knee allows the lower leg to rotate up until the point where the knee is locked out. In the same way the joints in the model limit the angle they can rotate to.

2.2.3 Collisions

A final constraint the limbs follow is that they collide with the floor. Each rigid body has an area, and the areas of the limbs and the area of the floor cannot overlap, otherwise the athlete would fall through the floor. For this reason the model needs to take collisions between the limbs and floor into account. The collisions between the limbs do not however need to be taken into account, as one limb can go behind another in the 2D view the user can see. To model this each rigid body can have a rectangular collision box, specified by a width and height. The boxes for the limbs can then be stopped from entering the box for the floor.

The QWOP world so far amounts to a set of rigid bodies and a set of joints on which the physics rules of planar rigid body dynamics can be applied. The set of rigid bodies will contain 13 in total; firstly a foot, lower leg, upper leg, lower arm and upper arm for each side of the body, and also a torso, head and finally the floor. Each of these have a width and height that define their collision box, a starting position, and a mass. The values of these variables are such that the floor is immoveable and spans the bottom of the world, and the attributes of the rigid bodies match the attributes of the limbs of the QWOP athlete. The set of joints contains 11 in total; a left and right heel, knee, hip, shoulder, elbow, and a neck. Each of these specify which two rigid bodies they join together, the offset from each of those bodies that the joint must stay, and the maximum and minimum angles between which the joints can rotate. These connect the limbs together in the right order and keep them together with the physical properties of an athlete.

This model so far delivers an athlete with limbs that can rotate freely, move in Cartesian space and has correct joint constraints and collision checking. The last thing to add is the

forces rubber?

2.3 Understanding Phaser

As decided in the Project Proposal, I used Phaser to create my JavaScript version of QWOP. One of the first things that I did to prepare was to start learning how Phaser works. Here I shall give a brief description of what Phaser is and how it works.

Phaser is a Desktop and mobile HTML5 game framework that is designed to take care of the mundane tasks involved with developing a game so that development can be sped up. It helps keep focus on the creative aspect of game design as time is not wasted on reinventing the wheel.

Phaser handles jobs such as creating the canvas the game will be drawn to, and provides frameworks that handle compatibility issues. For example it provides access

to simple methods for drawing graphics onto the canvas, and responding to user input, whether on a mobile device with a touch screen, or on a desktop with a keyboard and mouse.

To take advantage of these features, a developer first imports the Phaser library by adding the minified script to their web page.

```
<script type="text/javascript" src="js/phaser.min.js"></script>
```

It is then possible to start off ones JavaScript code with a line similar to this:

```
var game = new Phaser.Game(gameWidth, gameHeight,
Phaser.CANVAS, 'game-name', preload: preloadFunction,
create: createFunction, update: updateFunction, render:
renderFunction);
```

This code shows how the Game object is created. The Game object is the main controller for the entire Phaser game. First of all it handles the boot process. The boot process creates the canvas dimensioned to the width and height parameters provided and initialises Objects and data structures that will be used later. After it has initialised the game, control flow is passed to code the developer has written. Phaser simplifies controlling the execution flow over the lifetime of the game. It does this by giving the developer access to execute code at key moments in execution, so that the user doesn't have to write a lot of boilerplate code to organise this themselves.

The user specifies which code should be executed where using the JavaScript object passes as argument 4. One of its key-value pairs is `update:updateFunction` so the identifier `updateFunction` is given to be used for Phaser's `update` operation. This means that the contents of updateFunction will be run once every frame. In a similar way the developer can write a `preload` function that loads assets before the game starts that will be run at the beginning of the game.

2.3.1 Sprites

Another feature of Phaser is its use of sprites. In the Phaser documentation [1] sprites are described like so:

> 'At its most basic a Sprite consists of a set of coordinates and a texture that is rendered to the canvas. They also contain additional properties allowing for physics motion (via Sprite.body), input handling (via Sprite.input), events (via Sprite.events), animation (via Sprite.animations), camera culling and more.'

[1] definition from https://photonstorm.github.io/phaser-ce/Phaser.Sprite.html

2.3.2 Statistical tests

To prepare for the evaluation I investigated some statistical tests. Since lots of data is going to be collected in the evaluation, statistical tests needed to be compared to find appropriate ways of comparing this data.

For example to compare the distance reached for the two games, a t-test would be inappropriate as the participants are trying to go forwards and so the distances reached will likely not be from a Gaussian. since data will likely not be paired,

todo: keep writing this - add other ideas used in eval and how prepped

todo: make more graphs with the ordering like what Alastair said and then work out the t test and write that up in the prep.

Next I learnt how to do these thingys stuff in course stats test open coding?

In this chapter, I have discussed the theory that had to be understood, and work that had to be done, before implementation could begin

Chapter 3

Implementation

This section details the implementation of the mathematical model of QWOP designed in Section 2.2, and the implementation of the evaluation scaffolding used during the user study. Section 3.1 focuses on the implementation of QWOP (need to add what other subsections do). Section 3.2 explains the code written to produce and analyse the results of the user study. (again add subsections of this once done)

todo: rewrite chapter, fit with prep

3.1 Phaser Implementation of QWOP

Phaser is a JavaScript framework designed for making games. I have described how it works in the preparation section, and will now break down how it was used to make the different sections of the game.

3.1.1 Modelling joints in physics

The first hurdle was to work out how to create a body in Phaser's physics engine. 3.1 shows my first prototype for modelling joints and limbs, each rod is a sprite, and the joints between them are fixed together by Phaser's 'revoluteContraint's. This allows them to swing freely.

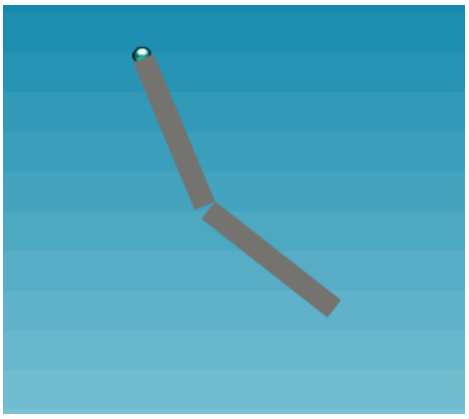

Figure 3.1: Swinging rods initial model of limbs

The next step was make the joints apply forces to each other. This would be used to not only move the limbs when the user presses the keys, but also provide forces so that the body would stay in a similar position when keys are not pressed.

The way I did this was to use Phaser's motors on the joints. These work by applying the force needed to move the joint at the specified speed, capped off at a max value. This way of powering joints could also be used easily to apply the static forces by setting the speed to 0.

3.1.2 Modelling the athlete

To model the athlete from these sprite joints, I specified the dimensions, mass, offset and starting rotation of each limb. Then forward kinematics is used on this data to get the starting points of the limbs. The limbs are then instantiated in these starting positions with the correct mass, dimensions and 'revoluteConstraint' joints. These joints are given joint limits so that each limb has a realistic range of motion, for example the knee joint doesn't let your lower leg rotate all the way round. The motors for these joints are then attached to the user input according to the control scheme. If a key is pressed that powers the limb, the motor is set to the limb power, otherwise it is set to 0 so that the static forces apply.

Each limb is given a group so that materials either part of the or group, that

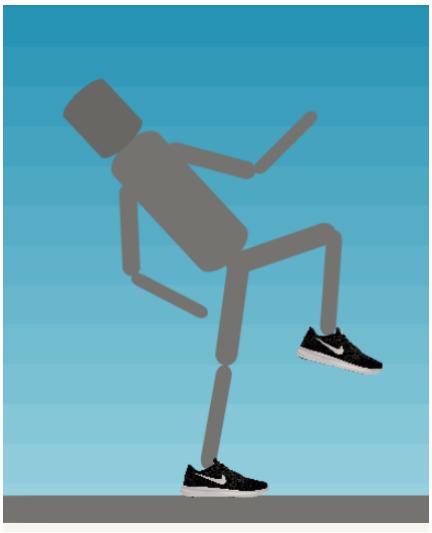

Figure 3.2: Model of athlete

3.1.3 Graphics

Next the graphics were added to the model so that it looked similar to the original QWOP. Each sprite was given an image taken from the original.

3.1.4 Game flow

Finally all the other components of the game were added:

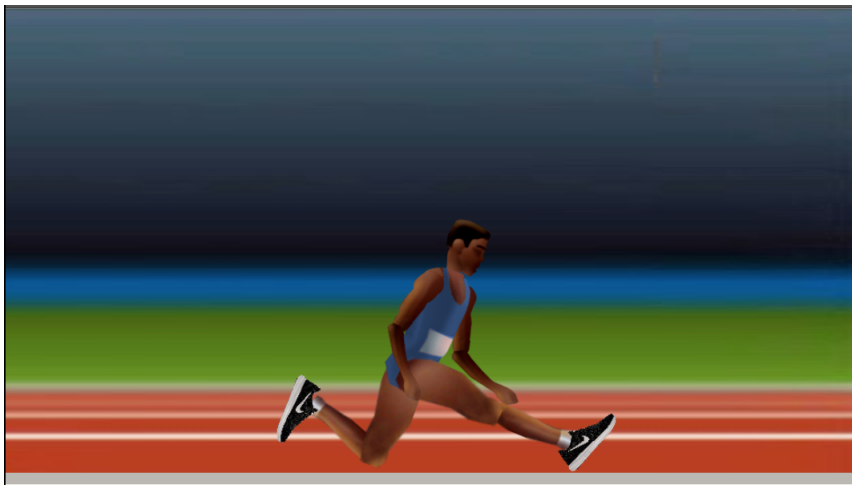

Figure 3.3: model with graphics added

1. Collisions with the floor - Each limb that cannot touch the floor is put into a collision group. This group is used to check whether a collision with the floor occurs, and if so end the game.

2. Calculation of distance reached so far - a function of the x value of the athletes torso

3. Text showing the distance reached on the screen.

4. Recording the high score and displaying it on the screen

5. A User help box shown on game start-up

6. Checking whether the user has reached the end of the 100m and displaying the congratulations message if the player gets to 100m

3.2 Implementation of evaluation software

To evaluate the project, I will be analysing the results from a user study. This analysis will be happening in the evaluation section, however a large amount of work went into implementing the testing environment so I will describe that here.

The center of the study is a controlled experiment, in which participants are randomly placed into one of two groups, each of which play the two versions of QWOP in a different order. Each game is played for 5 minutes while the key presses and distances reached for each participant are recorded. Either side of the experiment there is a questionnaire. Next I will explain the implementation of the evaluation scaffolding used during the user study.

3.2.1 Web page hosting user study

To host the instrumented software I wrote a webpage that guides the users through the sections of the study. As I wanted to get lots of participants to take part in my user study,

I wrote a web page to host the study so that participants could take part concurrently. It was also useful to host it on a web page because I can host both versions of QWOP in iframes, embed Google forms as questionnaires, and record the actions of the user. The actions of the user such as key presses and distances reached over time can be analysed along with other data to evaluate the success of the game.

images of page

3.2.2 Server software to receive and store user study data

I hosted the web page page on my SRCF web space. When the web page is recording data, it sends the data it has recorded to a PHP script at regular time intervals. This script records the data into separate files for each participant, so that all the data can be accessed in one place afterwards.

3.2.3 Chrome extension for reading distance from QWOP

Later on in the implementation I discovered that I couldn't access the distance that the player had reached for the original QWOP. It it impossible to access data from the game when hosting the original website inside an iframe, since all the game data is inside an anonymous function. Unfortunately When attempting to run the game by taking the html, JavaScript and other local resources that it accessed, and running them myself, the game's canvas goes orange and the game doesn't start. Since running an edited version of the original website did not work, I tried accessing the colour values of the game's canvas, then I could use OCR to read the distance from the screen itself. Accessing the pixel values from the canvas inside the iframe didn't work as QWOP has a OpenGL setting that made the GLBuffer unreadable.s In the end however the distance was made accessible by creating a chrome extension that took a screenshot of the page and placed it in a canvas. Then the data collection page takes the section of the screenshot that contains the distance text, inverts the colours so that it's black text on a white background, and places it into a smaller canvas. Then a lightweight JavaScript OCR script called ocrad.js is used to to read the text. With a little editing of the script's output, it gave an accurate value.

Chapter 4

Evaluation

4.1 Success Criteria

The project proposal talks about the aims of the project like so:

> At the end of the project I expect to be able to demonstrate a game similar to that of the original QWOP [...] although I am not aiming to replicate the behaviour of the original QWOP exactly [...]
>
> The athlete should have the same physical qualities that make it seem human, and the game should have the same gameplay items that make it work. The Work to be done section breaks down how this will be done in more detail. I propose to test how successful the project is with a user study. Since the quality of a game is subjective, it is appropriate to do a user study to evaluate the project.

I shall now clarify these objectives of the project into a success criteria. The rest of the evaluation will be aimed at showing with evidence that these criteria have been met. The success criteria are:

1. My JavaScript version is faithful to the original QWOP; it contains the same mechanics, that gives the player a similar experience.

2. My version will be different in some aspects, to the improvement of the game.

Since measuring 'improvement' is subjective, enjoyment of the player is a good metric to use.

To evaluate the project I have performed a user study. Section 3.2 of the Implementation chapter shows the implementation of the code used in the user study. The rest of this chapter will go into detail explaining the design choices for the user study, and analyse its results.

The center of the study is a controlled experiment. Participants are randomly placed into one of two groups, each of which play the two versions of QWOP in a different order. Each game is played for 5 minutes while the key presses and distances reached for each participant are recorded. More detail and analysis of this data is in Section 4.3

Before the participants take part in the controlled experiment, they are asked to fill in a questionnaire asking for their demographics and previous experience of playing QWOP and other video games. After the controlled experiment, there is another questionnaire that determines the participant's opinion of the two versions they have played. Sections 4.2 and 4.4 show the design and results of these questionnaires.

4.2 Intro Questionnaire Responses

The first questionnaire asked for the demographics of each participant. 30 participants took part in the user study. This is a good number because... Figure 4.1 shows the age and gender of the participants.

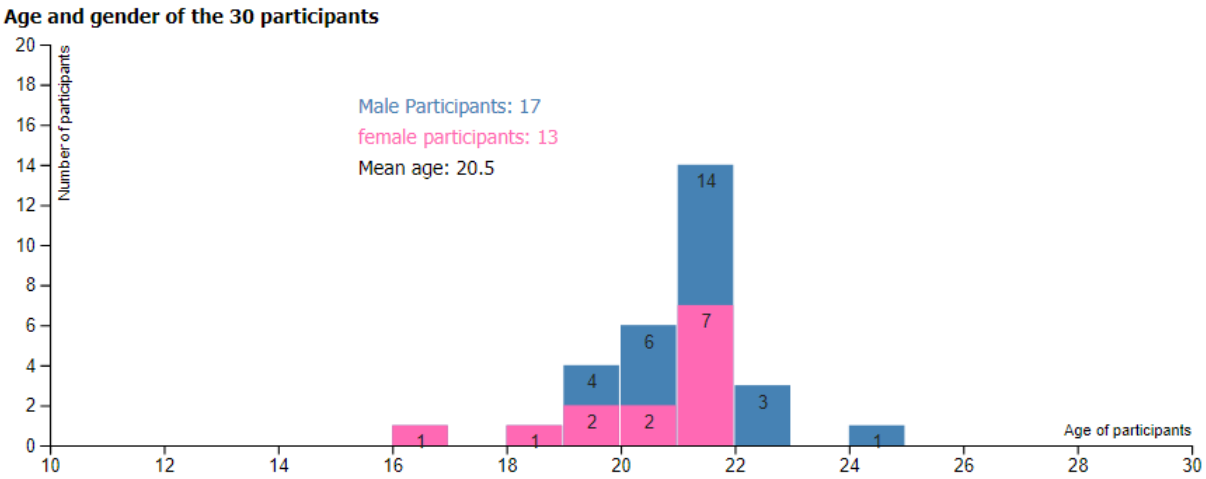

Figure 4.1: Participant demographics

The next question asked was whether the user had played QWOP beforehand, Figure 4.2 shows the options that were available to choose from and the frequency that the answers were picked. Most of the user hadn't played QWOP before. that matters because..???

Previous experience playing QWOP	Frequency of choice
Never played before	19
Less than 10 minutes	5
Between 10 and 30 minutes	3
Longer than 30 minutes	2
A long time over multiple sessions	1

Figure 4.2: Previous experience of participants

The third question asked for how long each participant plays video games during a typical week outside term time. Figure 4.2 shows the options that were available to choose from and the frequency that the answers were picked. This matters because..???

Time spent playing video games per week during a typical week outside of term time	Frequency of response
Less than an hour	22
Between 1 and 5 hours	2
Between 5 and 20 hours	4
More than 20 hours	2

Figure 4.3: Game playing experience of participants

4.3 Controlled experiment results

After the introductory questionnaire, each participant is randomly placed into one of the two groups, each of which play the two versions of QWOP in a different order. Each game is played for 5 minutes before the page moves onto the next section. The participant has 5 minutes to get as far as they can. Whenever they fall over they start again and can have as many restarts as they want in the 5 minutes. While each participant plays both versions, the keys being pressed and the distance reached is being recorded. Here I shall use this data to analyse whether my version is faithful to the original, and how they compare.

Figure 4.4 shows some examples of the distance recorded over time. Notice how the participants attempts to go forward, reach a distances, and then restart. Participant 2 reaches 85m at around 250 seconds.

To begin the analysis I shall compare the distances that the participants reach for each run; before they fall and restart or reach the end of the 5 minutes. Figure 4.5 shows a histogram of all of these restart distances over the 30 participants for my JavaScript remake, version gla. Figure 4.6 shows the same but for the original QWOP, version fod.

One striking fact is that the total number of restarts is so close. the number of runs for version gla is 924, and for version fod it is 925. My version is very faithful to the original in this regard and it shows that their gameplay share a property...

Version	Average time of run (s)
gla	9.74
fod	9.73

When comparing the restart distances of the two versions in Figure 4.5 and 4.6, the graphs share a similar shape. As they share the same area, 4.5 is a stretched version of 4.6

4.3.1 Investigation of Learning Bias

todo: this bit

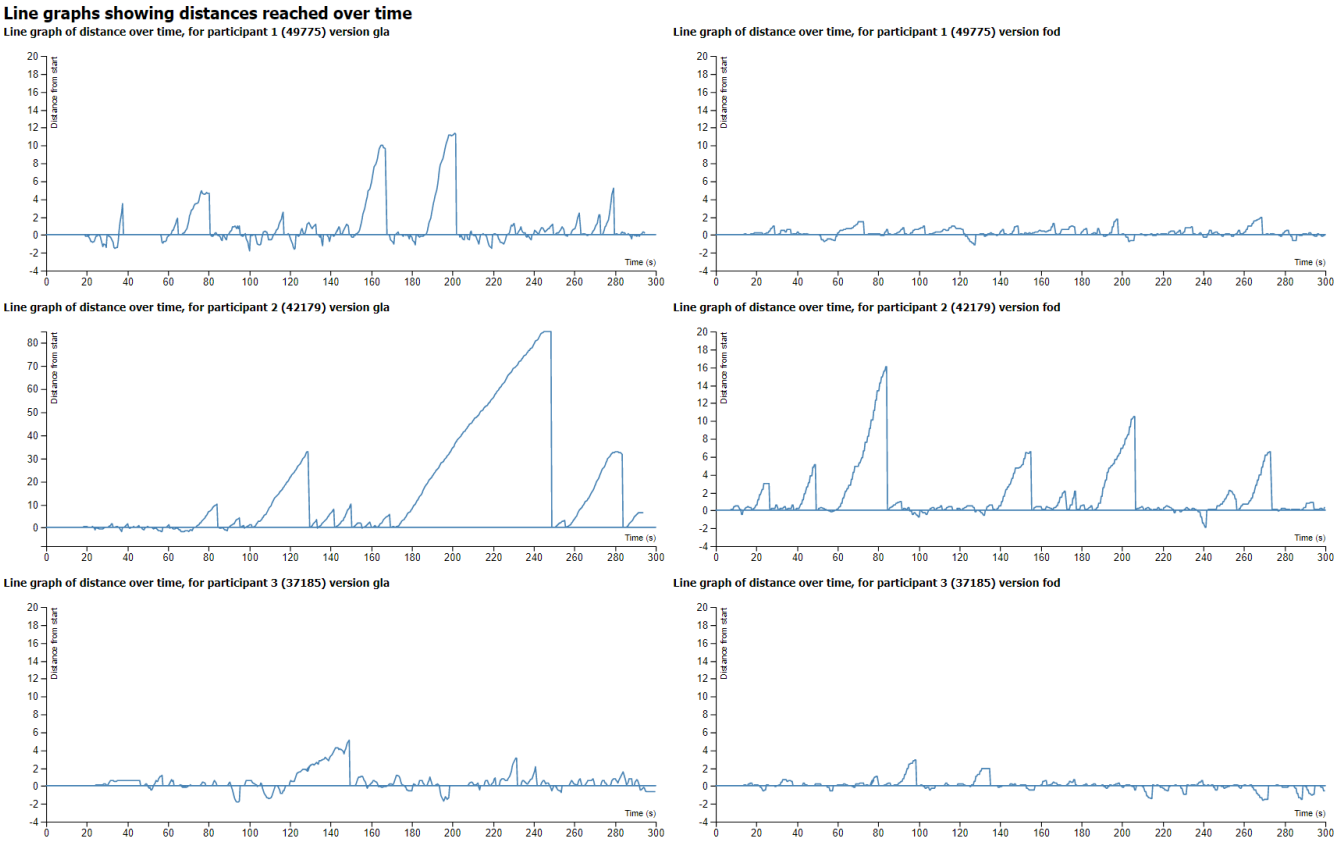

Figure 4.4: A selection of Distance over time graphs

4.4 Second Questionnaire

The second questionnaire happens after the participants have played both versions of the game. Here are the questions, with the figures containing the data. and it contains the following questions aimed at comparing the two versions:

1. Which version of the game do you think you were more successful at? Figure 4.8
2. Which version felt like you had more control over the athlete? Figure 4.9
3. Why do you think this was?
4. Do you think playing the first version of the game improved your performance in the second version? Figure 4.7
5. Which version did you enjoy more? Figure 4.9
6. Why do you think this was?

say about answers to choose from and why good from

Which version of the game do you think you were more successful at? could look at thought vs reality but doesn't seem that useful?

Which version felt like you had more control over the athlete? see Fig 4.9

why ? open coding?

first improved second?

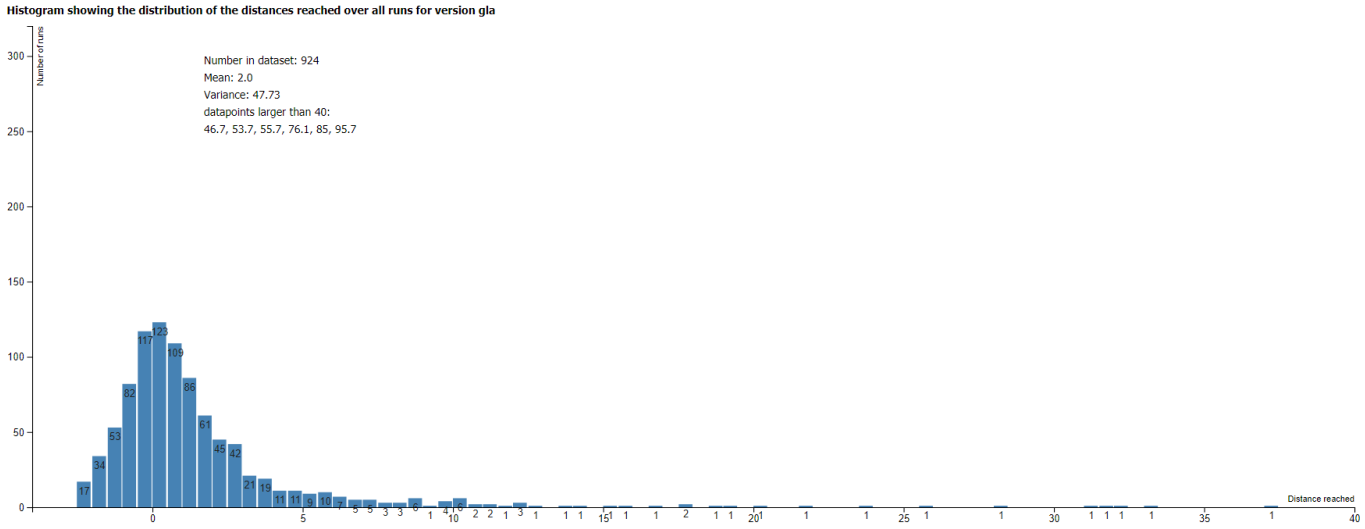

Figure 4.5: Restart distances for version gla

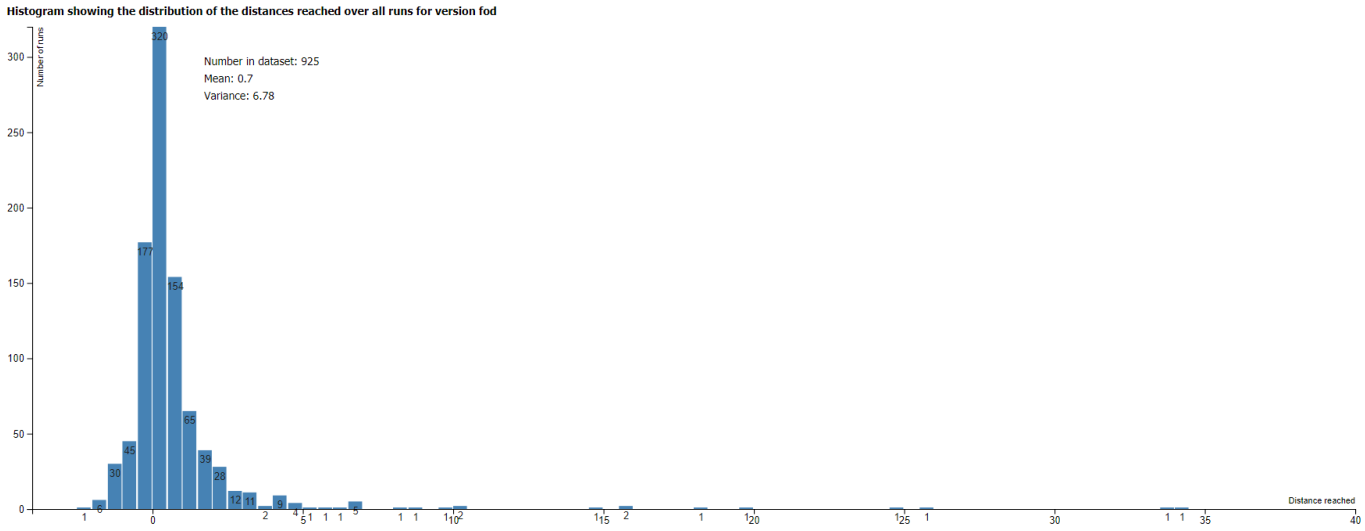

Figure 4.6: Restart distances for version fod

Opinion whether playing the first helped improve the second	Frequency of response		
	Group A	Group B	Total
Definitely	1	5	6
Maybe	5	7	12
I'm not sure	3	0	3
Maybe not	3	0	3
Definitely not	3	3	6

Figure 4.7: Participant opinion of whether playing the first game helped improve their score for the second, by group.

Version	Frequency of response		
	Group A	Group B	Total
First played	8	5	13
Second played	7	10	17
Version gla	8	10	18
Version fod	7	5	12

Figure 4.8: Participant opinion of whether they were more successful at the first or second version.

	Group A		Group B		Total over groups			
	First version (gla)	Second version (fod)	First version (fod)	Second version (gla)	gla	fod	First	Second
Which version felt like you had more control over the athlete?	7	8	4	11	18	12	11	19
Which version did you enjoy more?	9	6	5	10	19	11	14	16

Figure 4.9: Results of participant choice of version