

George Andersen

QWOP in JavaScript

Computer Science Tripos – Part II

Robinson College

May 4, 2018

Proforma

Name: **George Andersen**
College: **Robinson College**
Project Title: **QWOP in JavaScript**
Examination: **Computer Science Tripos – Part II, 2018**
Word Count: **5300¹**
Supervisor: **Dr Alastair Beresford**

Original Aims of the Project

Work Completed

Special Difficulties

None.

Declaration

I, George Andersen of Robinson College being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed George Andersen

Date May 4, 2018

¹Computed using TeXcount <http://app.uio.no/ifi/texcount/>.

Contents

1	Introduction	9
1.1	What is QWOP?	9
2	Preparation	11
2.1	Mathematical model of the QWOP world	11
2.1.1	Joints	11
2.1.2	Collisions	12
2.1.3	Forces	12
2.2	Understanding Phaser	13
2.2.1	Sprites	13
2.2.2	Physics engines	14
2.3	Statistical tests	14
3	Implementation	15
3.1	Phaser Implementation of QWOP	15
3.1.1	Modelling the athlete	15
3.1.2	Planar Rigid-Body Mechanics	16
3.1.3	Modelling joints in physics	16
3.1.4	Applying the joint forces	17
3.1.5	Graphics	18
3.1.6	Game flow	18
3.2	Implementation of evaluation software	19
3.2.1	Implementation of the webpage hosting the user study	20
3.2.2	Chrome extension for reading distance from QWOP	21
4	Evaluation	23
4.1	Success Criteria	23
4.2	Intro Questionnaire Responses	24
4.3	Controlled experiment results	26
4.3.1	Restart Distances	28
4.3.2	Investigation of Learning Bias	28
4.4	Second Questionnaire	29
5	Conclusion	35
A	Project Proposal	37

B	Questionnaire questions	39
B.1	First Questionnaire	39
B.2	Second Questionnaire	39

List of Figures

1.1	Starting screen of QWOP	9
1.2	10
3.1	Swinging rods initial model of limbs	17
3.2	Model of athlete	18
3.3	Model with graphics added	19
3.4	Diagram of user study flow	20
3.5	Opening page of the user study	21
4.1	Diagram of user study flow	24
4.2	Participant demographics	25
4.3	Previous experience of participants	25
4.4	Game playing experience of participants	26
4.5	A selection of Distance over time graphs. Participant 1 playing version gla, fod, and repeated for Participant 2.	27
4.6	A comparison of the distances participants reached over the two versions. .	31
4.7	Learning bias affecting version gla	32
4.8	Learning bias affecting version fod	33
4.9	Participant opinion of whether they were more successful at the first or second version.	34
4.10	Results of participant choice of version	34
4.11	Participant opinion of whether playing the first game helped improve their score for the second, by group.	34

Chapter 1

Introduction

The motivation for this project is to bring together multiple interesting areas of computer science. The project is to recreate an online game called QWOP using JavaScript, and to use a controlled experiment to test how it compares, in the participants view, to the original. It included implementing a physics simulation, reconstructing the graphics of the game, and investigating the Human Computer Interaction of playing online games. I decided to use JavaScript for the project as it was a good language for creating an online game, and I was interested in knowing in greater depth how web technologies fit together. The project would be a challenge as before starting the project I had very limited knowledge of JavaScript, and I would be carrying out a controlled experiment with a large number of participants. In the following section I shall provide an explanation of the game that I have recreated.

1.1 What is QWOP?

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, which sets the scene for the game and give instructions to the player of how to play.

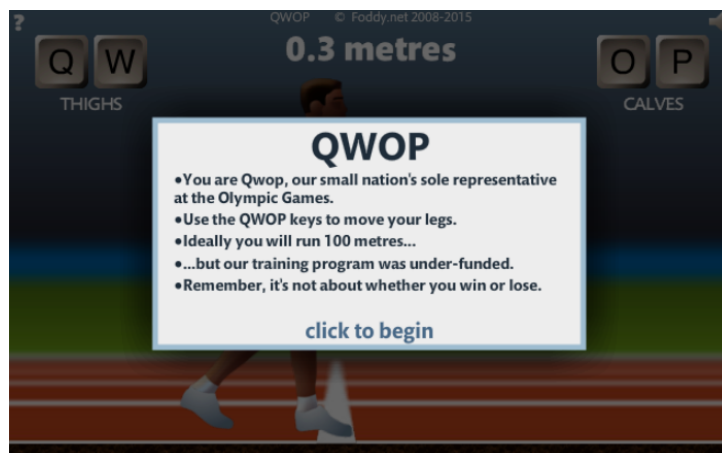
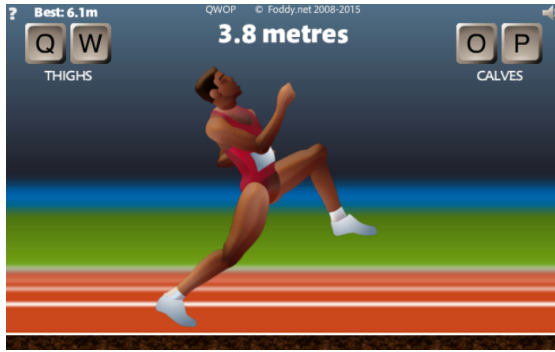
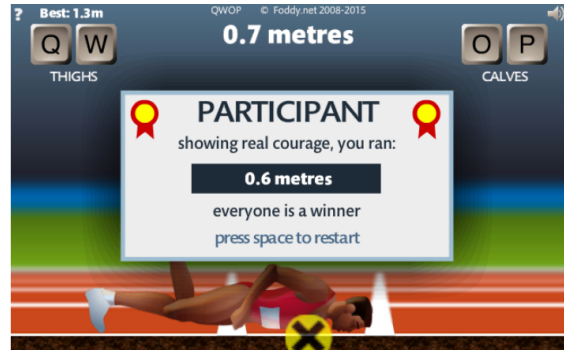


Figure 1.1: Starting screen of QWOP



(a) The athlete is running.



(b) The athlete has fallen over so the player must restart.

Figure 1.2

Players control the athlete using only the Q, W, O, and P keys. Whilst the Q key is held 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 then moves to begin they are free to press the keys to attempt to move forwards. Figure 1.2a shows the athlete running, and this invariably leads to Figure 1.2b which shows the athlete fallen over. Below the athlete there is an X symbol. This shows where the athlete first touched the floor with his arms or head. After this the player can press the space bar 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 some net movement forward. Both these methods are entertaining and typically lead to some success. Though the objective of QWOP is simple, it is notorious for being difficult to master ever since it went viral in 2010.

Chapter 2

Preparation

In this chapter I will discuss the preparation that happened before the implementation of the project began. In section 2.1 a mathematical model of QWOP is developed that is built in the implementation chapter. As suggested 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 learn how Phaser works. In section 2.2 I shall give a brief description of what Phaser is and how it works. Section 2.3 goes through the statistical tests that we're learned before the analysis in the evaluation could take place.

2.1 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. 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 system as a whole under the effects of gravity can be simulated. With only basic planar rigid-body dynamics, the limbs will simply fall to the floor. So next I will describe the required extensions to this model that emulate the behaviour of QWOP.

2.1.1 Joints

The members of an athlete are connected by joints. For example the foot and lower leg are connected at the ankle, and the lower leg and upper leg are connected at the knee. These constraints need to be captured in the model. For each joint-body connection, the joint stays in the same position relative to the frame of reference of the body – joints maintain a consistent offset from each connected member. For example, the ankle is

always at the back of the foot, and the ankle is always at the bottom of the lower leg. These two constraints work together in each joint to connect two limbs. In the same way, the mathematical model of QWOP will include joints that enforce these constraints. Another function of an athlete's joints is to limit the angles the limbs can rotate. For example the knee allows the lower leg to rotate up until the point where the upper and lower legs are in a straight line. In the same way the joints in the model limit the angle each rigid body can rotate.

2.1.2 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. 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 representing the floor. 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.

2.1.3 Forces

The QWOP world so far amounts to a set of rigid bodies and a set of joints on which the physics of planar rigid-body dynamics can be applied. There are 13 rigid bodies 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 immovable and spans the bottom of the world, and the attributes of the rigid bodies match the attributes of the limbs of the QWOP athlete. There are 11 joints in total; a left and right ankle, knee, hip, shoulder, elbow, and a single 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 move in Cartesian space whilst checking for collisions with the floor, and rotate freely whilst keeping joint constraints. A final addition to the model is a method of applying forces to the rigid bodies just as the muscles of an athlete would apply forces to its limbs. These include two types of forces, firstly the force when actively driving a limb forward. Secondly the forces that I will call static rotational forces. These static forces keep the body in the same shape when a limb is not being actively driven forward. For example if the player is not pressing any keys, then the athlete should hold his shape and start falling in whichever direction is natural, rather than collapsing to the floor.

2.2 Understanding Phaser

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 the developer 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, the event-based paradigm is used to pass control flow to code the developer has written. This simplifies controlling the execution flow by giving the developer access to execute code at key moments in execution called events. This is helpful as the developer doesn't have to write a lot of boilerplate code to organise this themselves. When one of these events arise, a callback function that the developer has written is executed. In argument four of the above code, the developer passes a JavaScript object to specify the callback functions for each event. One of the object's named argument pairs is `update:updateFunction`, so here the developer has given the identifier `updateFunction` 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 Phaser will run at the beginning of the game.

2.2.1 Sprites

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

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

“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.”

These are multi-purpose objects that are used to add components to a game. At base they are a set of x - y coordinates and a texture. For example to add a character to the game, one can choose an image to use as a texture, which can then be moved around the canvas by changing the coordinates of the sprite. Extra functionalities can be added to make use of other parts of Phaser, such as adding functions that handle certain inputs to the game, or adding animations.

todo: finish this

2.2.2 Physics engines

Phaser also comes with three different options of physics engines, “Arcade”, “Ninja” and “P2”. The difference between these options is that they start off simple and increase in complexity. This is useful as lightweight engines are preferred if the game is going to but run on a low-powered device such as a mobile phone. P2 is the most complex and has support for full body systems, including springs and collisions.

2.3 Statistical tests

To prepare for the evaluation chapter I investigated some statistical tests. Since lots of data is collected in the evaluation, statistical tests needed to be compared to find appropriate ways of comparing this data, for example comparing the distances participants reach over the two games. In this example a parametric test such as a t-test would be inappropriate, since the participants are trying to go forwards the distances reached will likely not be from a Gaussian or another symmetrical distribution.

The KolmogorovSmirnov test would be more useful in this case, as it gives a distance between. The data does not need to be paired as

since data will likely not be paired,

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

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 2.1, 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)

3.1 Phaser Implementation of QWOP

As in the project proposal, I have used Phaser to implement my version of QWOP in JavaScript. I provided an overview of what Phaser is and how it works in Section 2.2. I will now describe how it was used to make the different components of the game. First I shall show how it implemented the mathematical model of QWOP described in Section 2.1.

3.1.1 Modelling the athlete

The first task of my code is to create the model of the athlete, that is later built into the physics system. This starts with declaring the variables describing all the dimensions of the model. These variables lay out the dimensions of each limb, including their height, width, mass and starting rotation. They also specify the properties of the joints, for example the maximum and minimum angle that the joints can rotate to. Finally other values set forth other aspects of the model, such as the offset of the head from the top of the torso, or the distance between the left and right shoulder. These values specify the model of the athlete.

After these variables have been defined, a second set of variables are derived from them that will be used later. To do this forward kinematics is used on the previous set of variables to work out the starting coordinates of each limb. The code takes the starting position of the torso, and then uses the height of the torso and other offsets to get the x and y values for where the left upper arm should start. Then it uses the starting rotation, width and height of the upper arm to work out the starting position of the lower arm. This is repeated over all the limbs, until the x and y values of the starting positions are specified for all of the rigid bodies.

3.1.2 Planar Rigid-Body Mechanics

Now that the model of the athlete has been defined, the core of the mathematical model, planar rigid-body mechanics, is implemented. To implement the Planar rigid-body mechanics of Section 2.1, I used P2, one of the physics engines that comes with Phaser. P2 and other aspects of Phaser such as the game object have been described in Section 2.2. The next section of my code first runs the set-up described there, after which the P2 physics system is initialised with this code:

```
game.physics.startSystem(Phaser.Physics.P2JS);
```

P2 uses sprites, as I described earlier in section 2.2.1. One of their functionalities is that they can be added to one of Phaser's physics systems, meaning that their x and y values are controlled to follow the physics rules you set up. In this code, a sprite is created and added to the game object.

```
head = game.add.sprite(headX , headY , 'head');
```

The `headX` and `headY` values in the above code are the x and y values of the head defined earlier. Similar code is repeated over all of the x and y variables to create a sprite for each rigid body of the model.

These sprites are then enabled in the physics system using this code:

```
game.physics.p2.enable([torso, upperLegLeft, lowerLegLeft, shoeLeft,  
    upperLegRight, lowerLegRight, shoeRight, upperArmLeft,  
    lowerArmLeft, upperArmRight, lowerArmRight, head, floor],false);
```

This allows P2 to control their coordinates.

3.1.3 Modelling joints in physics

Figure 3.1 shows my first prototype for modelling joints and limbs. The rods are held together by joints and can swing freely. Each rod is a sprite, and each joint connecting them is the joint used to control rotations in P2 physics, called a **revoluteConstraint**.

The next stage of my code implements section 2.1.1 of the mathematical model, the joints between the rigid bodies. These joints are created between the sprites created earlier. The code below implements the left ankle, the joint connecting the left foot with the left lower leg:

```
ankleLeft = game.physics.p2.createRevoluteConstraint(shoeLeft,  
    [-shoeXOffset,-shoeLegDistance],lowerLegLeft,  
    [0,lowerLegHeight/2],maxForce);  
  
ankleLeft.upperLimit = ankleMaxAngle;  
ankleLeft.lowerLimit = ankleMinAngle;
```

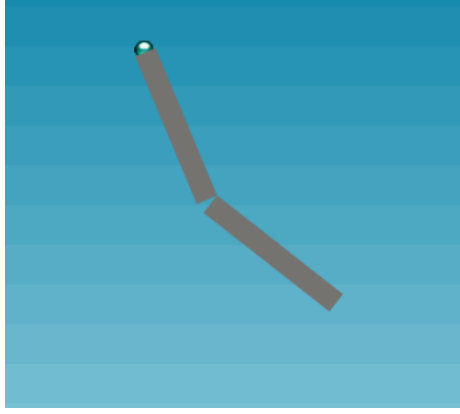



Figure 3.1: Swinging rods initial model of limbs

Firstly the `RevoluteConstraint` joint is created. The first and third arguments specify the sprites in the physics system that are to be connected together by the joint. In this case the left foot sprite is connected to the left lower leg sprite. The second and fourth arguments set the offset from the centre of each limb that the joint should stay. Here the ankle should always be at the top and towards the back of the foot, and it should be at the bottom of the lower leg, `lowerLegHeight/2` from the center of the lower leg.

Secondly the upper and lower limits of the ankle are set to stop the ankle rotating into impossible positions. Again the angle variables used are the variables that specified the model earlier, and similar code is repeated for all the joints in the model.

3.1.4 Applying the joint forces

In Section 2.1.3, the mathematical model of QWOP describes two types of forces used by the athlete. Firstly active forces that the athlete uses to drive a limb forward, and secondly static rotational forces that are used to keep the body in shape so that it does not collapse to the floor. In the previous section of code, the joints are created, and now they are used to implement these forces. The mechanism used to apply these forces is to use a set of methods build into Phaser's `RevoluteConstraint` object that add the functionality of a motor to the joint. The motor applies a force over its two sprites such that the angular velocity of the joint will increase or decrease towards the motor speed. This force it applies is limited by the maximum torque of the motor, which is set by the `motorMaxTorque` variable belonging to each `RevoluteConstraint` joint. The motor is turned on and off with `enableMotor()` and `disableMotor()`, and the speed set with `setMotorSpeed()`. Both types of forces are applied using the same method, as if the Motor speed is set to a positive value, the motor will increase the angular velocity to that value, which drives the limbs forward. Whereas if the motor speed is set to 0, the motor will apply the necessary forces to stop the joints from rotating so that the athlete stays in shape.

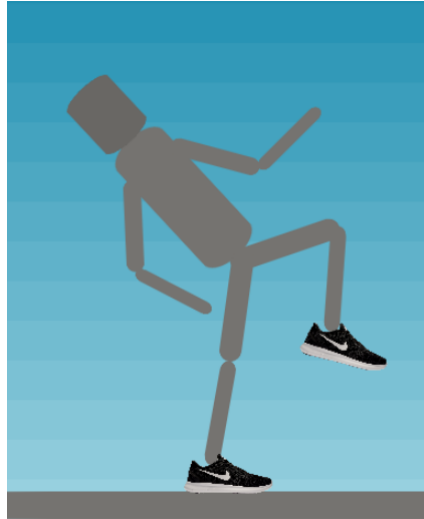


Figure 3.2: Model of athlete

3.1.5 Graphics

Once I had implemented the mathematical model of QWOP in the physics simulation, the next stage was to change the graphics so that it looked similar to the original QWOP. To do this, I used the Chrome developer tools to access the files that the original QWOP downloaded when loading. In these files I found an image containing the graphics for the athlete, and other images for other components such as the background and help screens. To add these graphics into the game, I separated the textures into separate images, and set them as the textures for the corresponding sprites. Figure 3.3 shows the implementation of the model after the graphics had been added.

3.1.6 Game flow

After the core components of the game were implemented, I added the other components needed to emulate the original QWOP. Firstly, calculating the distance the player has run so far. To do this I wrote a function that took the x value belonging to the athletes torso in the physics simulation, and returned a distance in meters. The function calculates the distance between the athlete and the starting line, and scales it such that it is the same distance the original judges you to have reached, for moving the same distance along the screen.

Other components added include the text which updates to show the distance the athlete has reached, the ability to record a high-score the user had got, and text to display the current high-score on the screen. The text that shows how far you have reached updates ever frame, and is changed in the `update` function. The way Phaser controls execution flow using callback functions such as `update` is described in Section 2.2. Here is an excerpt of the `update` function showing the text being updated each frame of the game:

```
// Update Score Text
currentScore = distanceTraveled();
```



Figure 3.3: Model with graphics added

```
scoreText.text = currentScore + " meters";
// Check if the player has won the game
if (gameOver == false && currentScore>100) {
    gameOver = true;
    coverSprite = game.add.image(gameWidth/2,gameHeight/2, 'gameWon');
}
```

The second function of this code is to check whether the athlete has reached the end of the 100m. If so it displays the congratulations message and allows the player to restart. The Final components I added were the help boxes that are shown when the game begins, when the athlete falls over, and when the game has been completed as in the code above.

3.2 Implementation of evaluation software

To evaluate the project, I will be analysing the results from the user study. The evaluation chapter contains this analysis, however a large amount of work went into implementing the testing environment, so I will describe that here. Figure 3.4 shows the sections of the study.

First of all the participants are briefed on what the study will contain, and are informed of their ability to opt out whenever they wish. If a participant decides to opt out, their data will be deleted. After participants have given their informed consent to take part in the study, they move on to the rest of the study, which is hosted on a webpage. The first section asks the user to fill in a Google form. This first questionnaire finds out the demographics and previous experience of each participant. The main component of the study follows. In a controlled experiment, each participant is randomly placed into one of two groups. The two groups play both versions of QWOP in a different order. Each game is instrumented, so that while the participants play, their key presses and the distance reached down the track are recorded. Both versions are played for 5 minutes before the study moves onto the next section. After the users have used the instrumented software, they fill out a second questionnaire that records the opinions of the participant through

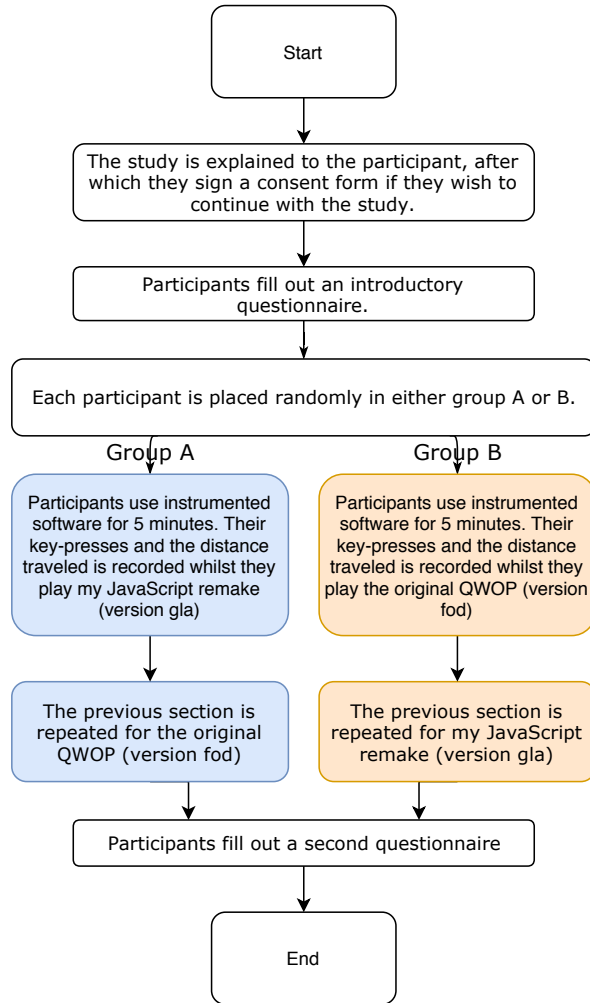


Figure 3.4: Diagram of user study flow

questions such as “Which version felt like you had more control over the athlete?” and “Why do you think this was?”.

The Evaluation chapter goes into detail explaining the design choices for the user study, and analyses its results. In the next section I will explain the implementation of the code and the other elements of the evaluation scaffolding used to facilitate the user study.

3.2.1 Implementation of the webpage hosting the user study

To host the user study, I wrote a webpage that guides the users through the sections of the study, shown in Figure 3.4. The webpage consists of an HTML page and JavaScript file that controls the use of the page. The HTML page contains elements for the page title, multiple elements for text, a `Next` button, and an `iframe` that hosts another website. The button is used to move to the next section of the study. When clicked, the contents of the other elements are updated to construct the next section of the study. The text gives instructions for how the participants should continue, and the `iframe` contains one

of the questionnaires or one of the instrumented versions of QWOP.

Hosting the study on the webpage allowed participants to take part in my user study concurrently, which helped me maximise the number of people taking part in the study. Other benefits of hosting the user study on a webpage were that it was simple to embed Google form questionnaires, and I could host instrumented adaptations of both versions of QWOP to log data from the experiment.

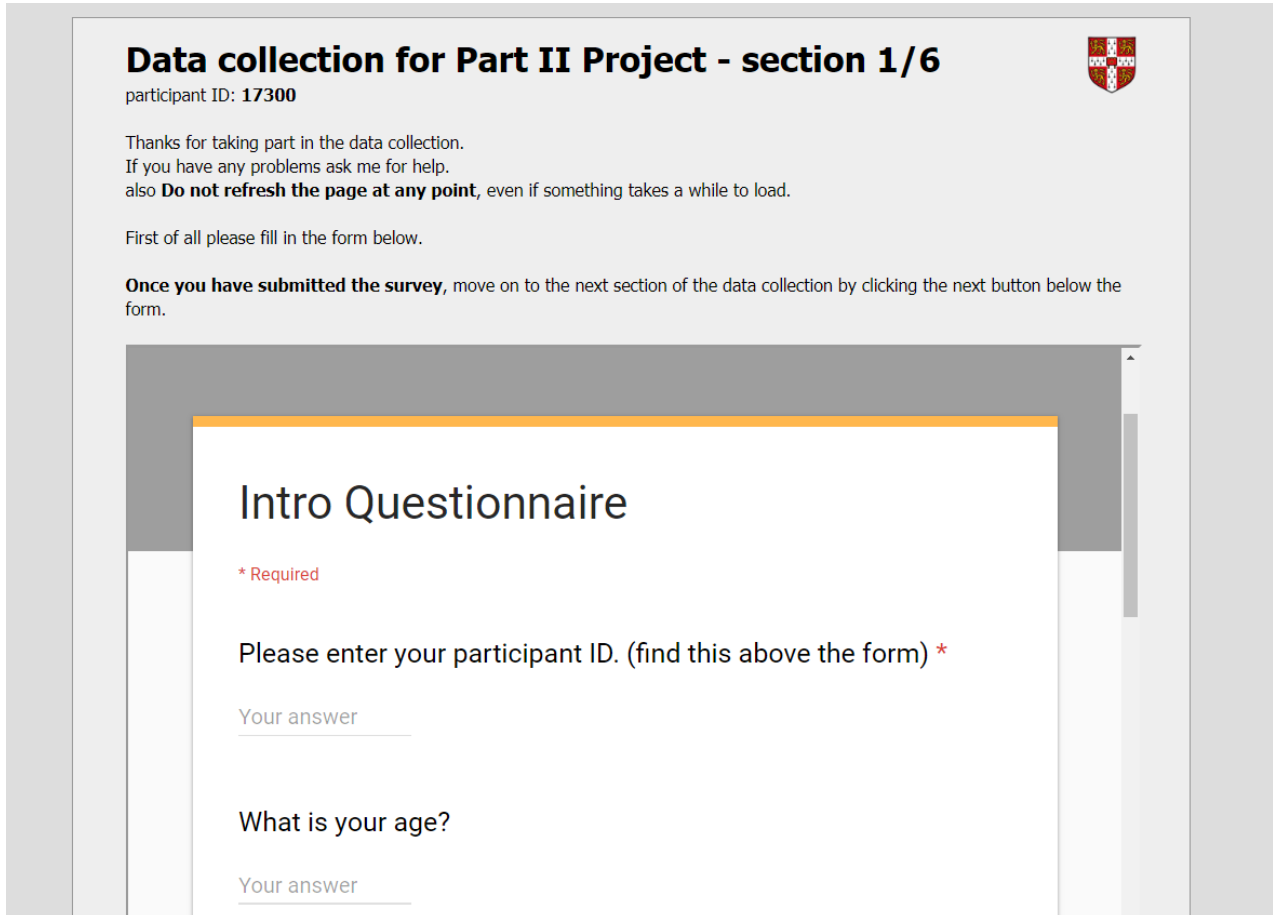
The image shows a web browser window displaying a data collection form. The title is "Data collection for Part II Project - section 1/6" in bold black text. Below the title, the participant ID "17300" is displayed. A small red and white crest logo is in the top right corner. The text "Thanks for taking part in the data collection. If you have any problems ask me for help. also **Do not refresh the page at any point**, even if something takes a while to load." is shown. Below this, it says "First of all please fill in the form below." and "Once you have submitted the survey, move on to the next section of the data collection by clicking the next button below the form." The main content area is titled "Intro Questionnaire" and contains two required questions: "Please enter your participant ID. (find this above the form) *" and "What is your age?". Each question has a "Your answer" input field below it. The form is set against a light gray background with a white content area.

Figure 3.5: Opening page of the user study

I hosted the web page on my SRCF web space. Whilst the web page is recording data, it logs the data by sending it to the server at regular time intervals. I wrote a PHP script that the server uses to process this data server-side and record it into separate files for each participant. This creates a JSON database of all the key-presses and distances travelled for the participants, which means that a group of participants can take part in the study, and then afterwards all the data is easily accessible server-side.

3.2.2 Chrome extension for reading distance from QWOP

Later on in the implementation I encountered a problem when instrumenting the original QWOP. I discovered that I couldn't access the distance that the player had run down the track. To host the original on my data collection page, the website for the original was placed in the iframe. This let the user play the game, although it was impossible for my

page to access any internal data from the outside, since all the game data is inside an anonymous function.

To instrument my version I added code to log the key-presses and send the key log buffer and current distance to the server at regular intervals. I inserted similar code into the iframe of the original to record key-presses, however since the internal data of the game was not accessible, I had to find another method of accessing the distance the player had reached.

Another method I attempted was to take the HTML, JavaScript and other local resources that the game accessed, and run them locally. In this way an edited version could be run that outputs the distance travelled. This did not work as the game appears to run some counter hosting measures such that the game does not start, and instead, the canvas goes orange. Since this did not work I tried accessing the colour values of the game's canvas, so that I could use OCR to read the distance from the screen itself. This did not work however as accessing the pixel values from the canvas inside the iframe was blocked. The original QWOP has a OpenGL setting that made the GLBuffer unreadable. In the end I solved this problem by creating a chrome extension that took a screenshot of the page over regular intervals so that OCR could be used. The extension used it's developer privileges to take a screenshot of the page and place it in a canvas. The data collection page then locates the text in the screenshot that shows how far the player has run, and uses a lightweight JavaScript OCR script called `ocrad.js`¹ to read the distance the athlete has travelled. The script gave accurate results with a some editing of the output after the colours of the input image are inverted so that it's black text on a white background.

¹`ocrad.js` is available at <http://antimatter15.com/ocrad.js/demo.html>

Chapter 4

Evaluation

4.1 Success Criteria

Referring back to the project proposal, I proposed to evaluate the project with a user study:

“At the end of the project I expect to be able to demonstrate a game similar to that of the original QWOP. 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...”

In order to test the success of the project, I refined the success criteria. The rest of the evaluation, including the user study will provide evidence that these criteria have been met. The success criteria are:

1. I have developed a game similar to that of the original QWOP in JavaScript.
2. My version is faithful to the original QWOP; it contains the same mechanics and provides a similar playing experience.
3. My version aims to be more playable for a typical player and I will quantify the improvement in a user study.

Previously, the Implementation chapter described the components of the user study. Figure 4.1 shows how a participant progresses through these sections. Section 3.2 of the implementation chapter explains the implementation of the system used to facilitate the user study, including the code written for the webpage, and how the games were instrumented. The rest of this chapter will explain the design choices for the user study, and analyse its results.

After participants have given their informed consent to take part in the study, they move on to the first questionnaire, which is hosted on the data collection webpage. In section 4.2 the results of this first questionnaire are analysed.

The main component of the study follows, the controlled experiment. In Section 4.3, the controlled assessment will be explained, and the data recorded during the controlled experiment will be presented and analysed.

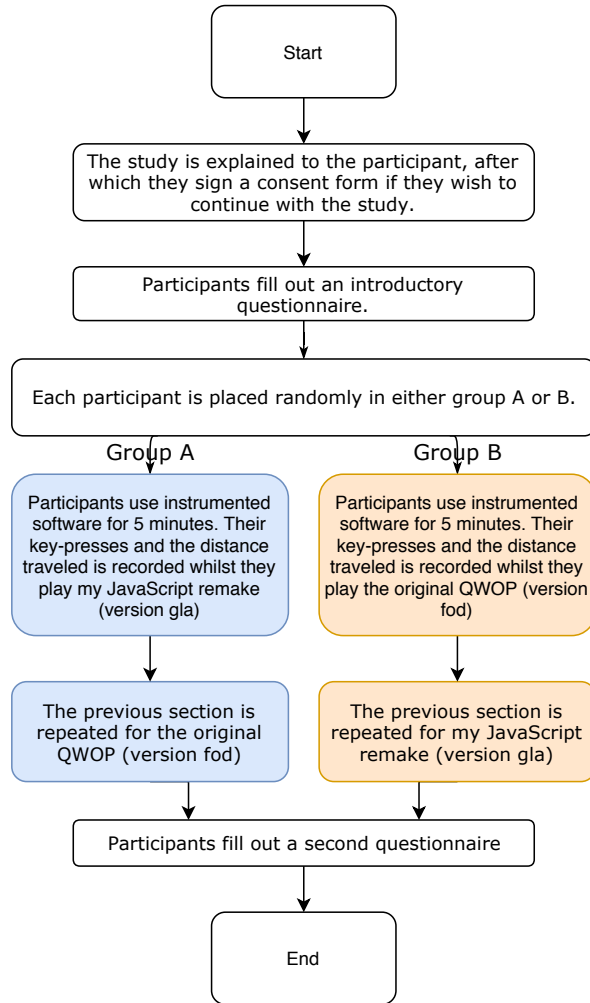


Figure 4.1: Diagram of user study flow

After the users have used the instrumented software, they fill out a second questionnaire that records the opinions participants have of the two versions. The results of this questionnaire are given in section 4.4.

4.2 Intro Questionnaire Responses

The participants that took part in the study were students, some of whom I recruited at the Computer Laboratory. I set up the user study software on a few computers in the Intel Laboratory and recruited people in person and through social media. The first questionnaire asked for the demographics of each participant. Figure 4.2 shows the age and gender of the 30 participants that took part in the study.

In order for my controlled experiment to have good external validity, it was important that the recruited participants were representative of the population for which I want to make conclusions. According to a study by the Pew Research Center¹ younger people and

¹<http://www.pewinternet.org/2008/12/07/adults-and-video-games/>

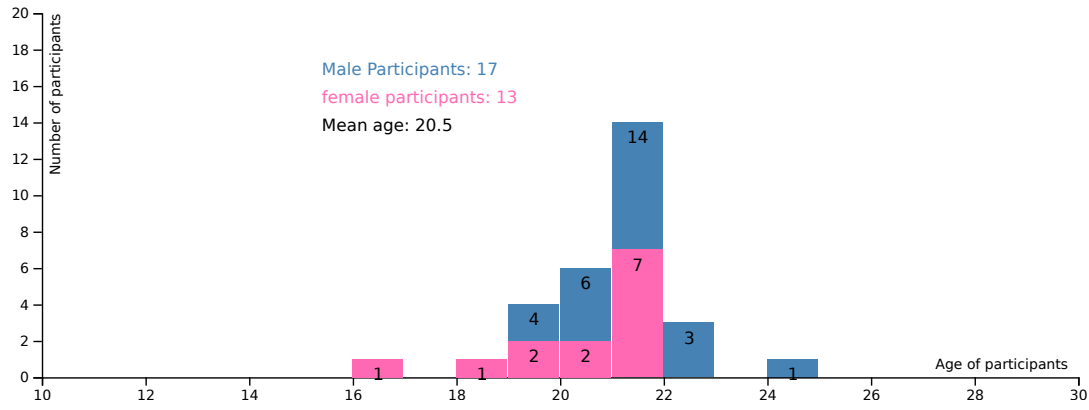


Figure 4.2: Participant demographics

students are more likely to play games than others:

“Independent of all other factors, younger adults are considerably more likely than older adults to play games, and the likelihood that an adult is a video gamer decreases significantly with age.

A person’s education level is another predictor of video game play. Some 57% of respondents with at least some college education play games, significantly more than high school graduates (51%) and those who have less than a high school education (40%). Current students who are 18 or older are also avid players.”

The Evaluation Criteria show that the study aims to make conclusions about a typical QWOP player. Since the recruited participants were younger people and students, they represent a typical game player and hence QWOP user. So in this regard the study has good external validity.

The next question of the questionnaire asked whether the user had played QWOP beforehand. Figure 4.3 shows the options that were available to choose from and the frequency that the answers were picked.

Previous experience playing QWOP	Frequency of response
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 4.3: Previous experience of participants

The majority of the participants had not played QWOP before. For those who had, only one had played the game for a long time previously.

As the participants only play each game for 5 minutes, and as there were only a few participants who had played the game before, the experiment does not provide enough

data to comment on whether the game fulfils the success criteria for players who play the game for some time. Because of this it may seem that the experiment is not ecologically valid, as the real life use case of playing the game over a longer period of time is not experimented on.

However, because of the nature of online games, the first impressions of a game are very important as if the player gets bored they can just click on another game. As such a typical player will be one who enjoys the first chunk of time playing the game.

The third question of the introductory questionnaire asked for how long each participant plays video games during a typical week outside term time. Figure 4.3 shows the options that were available to choose from and the frequency that the answers were picked.

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 4.4: Game playing experience of participants

4.3 Controlled experiment results

In this section, the controlled assessment will be explained, and the data recorded during the controlled experiment will be presented and analysed.

During the controlled experiment each participant was randomly placed into one of two groups. The two groups played both versions of the game with instrumented code added. This allowed me to record the keys that the participants pressed, and the distance that they travelled over time. The implementation of the instrumented software is described in the Implementation chapter, Section 3.2.

Group A played my version (which I shall call version gla) first and then played the original (version fod, after its creator Bennet Foddy). Group B played them in the opposite order. Each game was played for 5 minutes. When a participant moved on to a section of the user study where one of the games is played, a timer is started. When the 5 minutes is up, the page moves onto the next section. Between the two games there is a break page where the participant can press next when they are ready for the second version. Participants have these 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. Figure 4.5 shows examples of the distance some of the participants reached recorded over time.

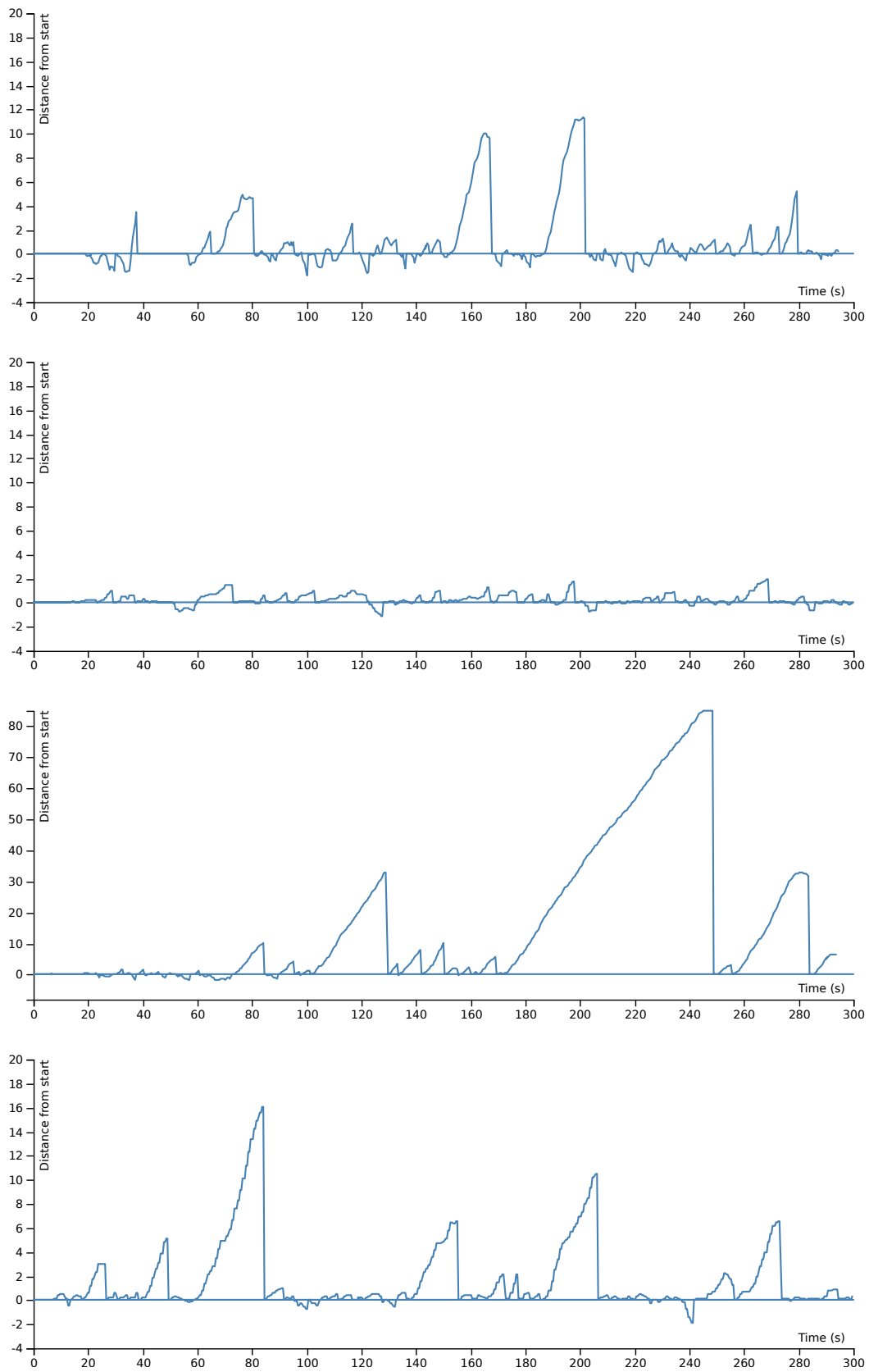


Figure 4.5: A selection of Distance over time graphs. Participant 1 playing version gla, fod, and repeated for Participant 2.

4.3.1 Restart Distances

Figure 4.5 shows some examples of the distance participants reached throughout the 5 minutes they played the version for. One can see the participants attempt to go forward, reach a distance, and then restart. To begin the analysis of this data I shall compare the distances that the participants reach at the end of each run; before they fall and restart, or run out of time. I will be calling these distances restart distances.

Figure 4.6a shows a histogram of all of these restart distances over the 30 participants for my JavaScript remake, version gla. Figure 4.6b shows the same but for the original QWOP, version fod. These histograms contain all the restart distances for each version, and includes both participant groups. The affect of the order a participant played the games is analysed later, as Section 4.3.2 investigates the learning bias of playing the other version first.

Figure 4.6a and 4.6b can be compared to see the change in the restart distances for the two versions. The graphs share a similar shape. The total number of runs participants made when playing version gla is 924, and for version fod 925. This means that firstly, both graphs have almost exactly the same area, even though they look quite different. Secondly it shows that the average time a run lasted over all participants is very similar. My version is very faithful to the original in this regard, and this data is evidence that success criterion 2 – having a similar gameplay experience – is met.

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

Another piece of evidence that supports criterion 2 is that both graphs have a similar shape. Both look like they belong to the Gaussian family of curves, although they are not symmetrical, as the athlete cannot go back further than the back wall. The fact that participants fall backwards as often as they manage to go forward in both versions is a testament to the difficulty of controlling the athlete. The curve for version gla in Figure 4.6a looks like a stretched version of the curve in Figure 4.6b. This shows that it was easier for participants to reach further in version gla. This can be seen in the increase of the mean between the two versions, from 0.7 to 2.0, but more clearly in the increase of variance – from 7 to 48 – and in the maximum values the participants reached.

Even though the track the athlete has to run down is 100 meters, the x -axis of the graph stretches until 40m so that the important section of the graph can be seen more clearly. No participant managed to go further than 40m for version fod, and the values above 40m for version gla are stated above the graph.

4.3.2 Investigation of Learning Bias

When the participants started the user study, they were given a randomly generated participant ID. This number was used to link the questionnaire responses of each participant to the data logged during the controlled experiment. This number was also used to split

the participants into the two groups for the A/B testing. The benefit of using this random number to make the experiment a randomised control trial is that selection bias is minimised.

Another benefit of this experimental design is that the effect of playing one game before the other can be taken into consideration and not affect the analysis.

Figures 4.7 and 4.8 illustrate this learning bias affecting the participants. For example in 4.7, the learning bias for version gla can be seen. The blue and khaki histograms both show the restart distances for the participants when playing version gla, however the blue histogram shows the restart distances for the participants in group A and the khaki for participants in group B.

The participants in group A play version gla first, whereas the participants in group B play version fod first, and so group B have the advantage of playing version fod beforehand. This learning bias can be seen in Figure 4.7 to have a small effect on the performance of participants playing version gla. There is a small increase of the number of attempts from 427 to 497, and these are mostly manifested between 0 and -1, otherwise the curves are very similar. This seems likely down to chance as there would be no reason for preceding play time of version fod to cause participants to fall over backwards more often.

Figure 4.8 takes the same approach but for version fod. Group B played version fod first, and group A played fod after playing version gla. Again the layered histograms visualise the effect that playing the other version beforehand has, showing the group with the learning bias in Khaki. This time there is a different effect. Learning from version gla reduced the number of attempts made by 15 participants from 530 to 395.

4.4 Second Questionnaire

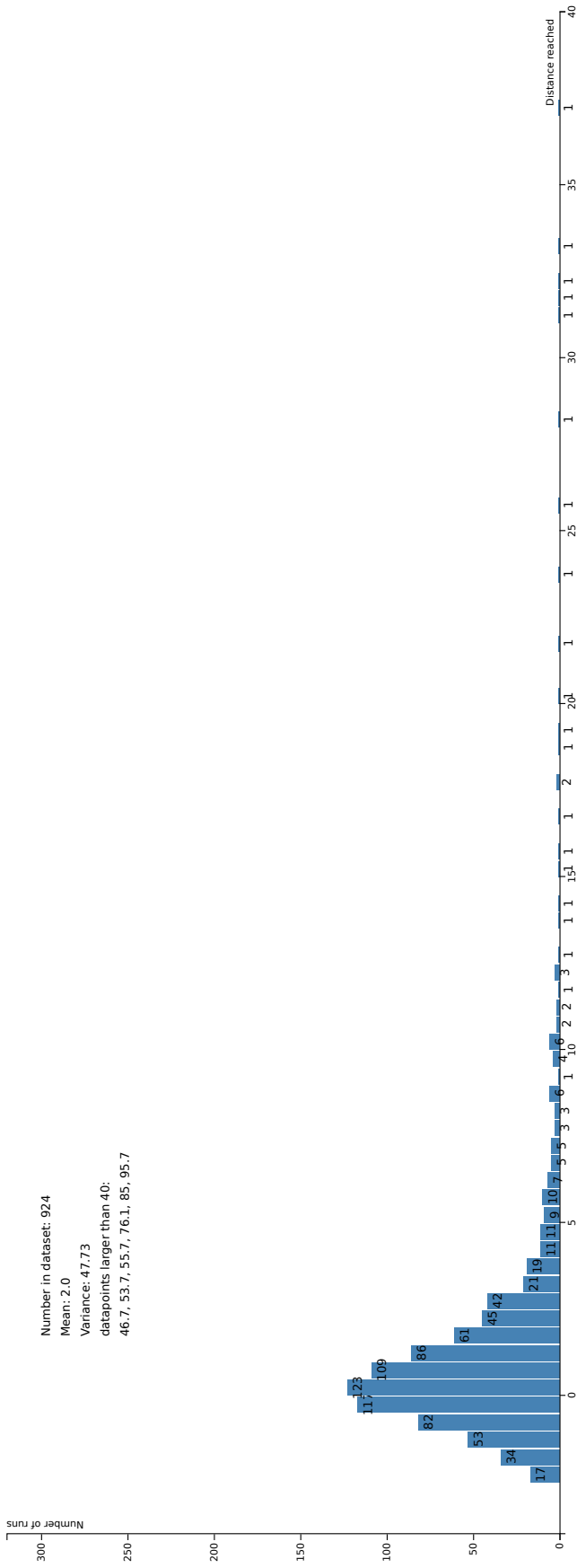
Participants complete the second questionnaire after they have played both versions of the game. It contains questions aimed at comparing the two versions. The questions and the answers participants could pick from are available in Appendix B.2. This part of the evaluation analyses the participants views of the different versions.

Figure 4.9 shows the results for the question “Which version of the game do you think you were more successful at?”. For this question the answers the participants had to choose from were “The first version” and “The second version”. This removes bias towards either version compared with giving the participants a choice between the two versions identified by their differences. The different participant groups played the games in a different order, so 4.10 shows the results that the participants gave, and how they translate into which version participants thought they were more successful at overall.

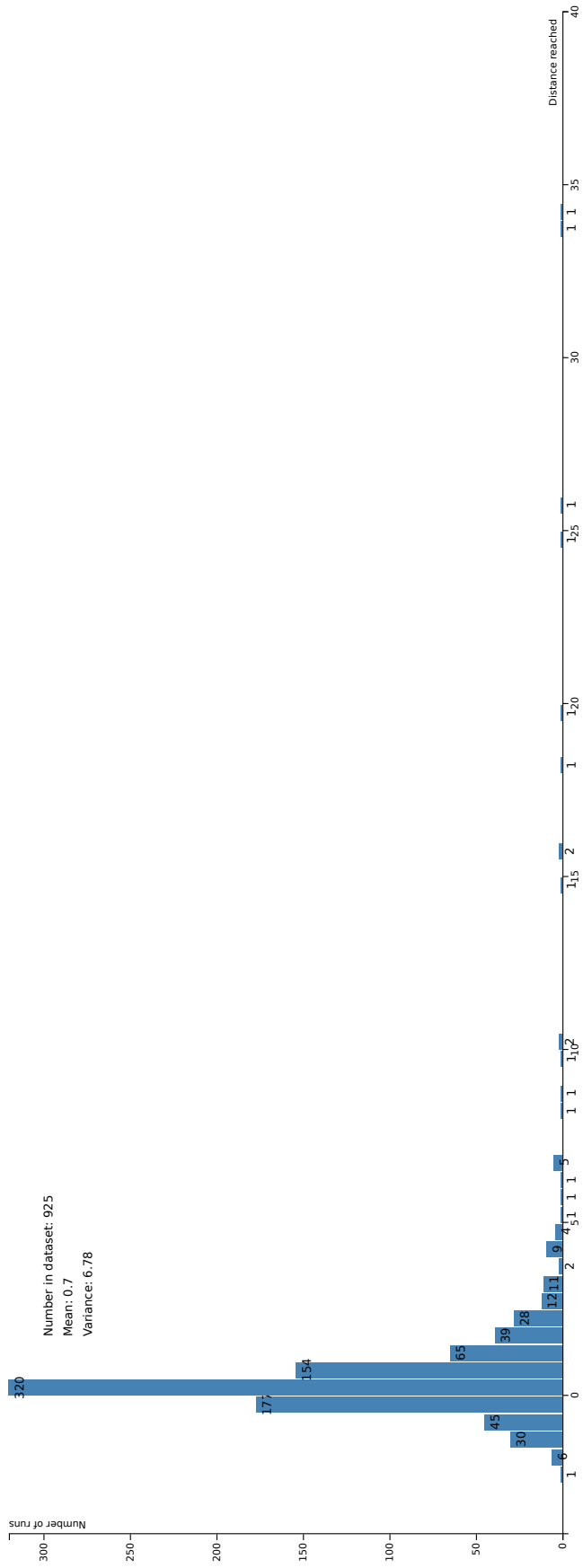
Figure 4.10 contains the data of the responses to the question “Which version felt like you had more control over the athlete?” and “Which version did you enjoy more?”. Similar to the question above, both of these questions gave the choice between the first and second version. Taking into account the different participant groups, 19 of the 30 participants said they found version gla to be more enjoyable than version fod. This is strong evidence for the third success criterion:

“My version aims to be more playable for a typical player and I will quantify the improvement in a user study.”

Figure 4.11 gives the responses to the question “Do you think playing the first version of the game improved your performance in the second version?” in relation to the bias section earlier



(a) Restart distances for version gla



(b) Restart distances for version fod

Figure 4.6: A comparison of the distances participants reached over the two versions.

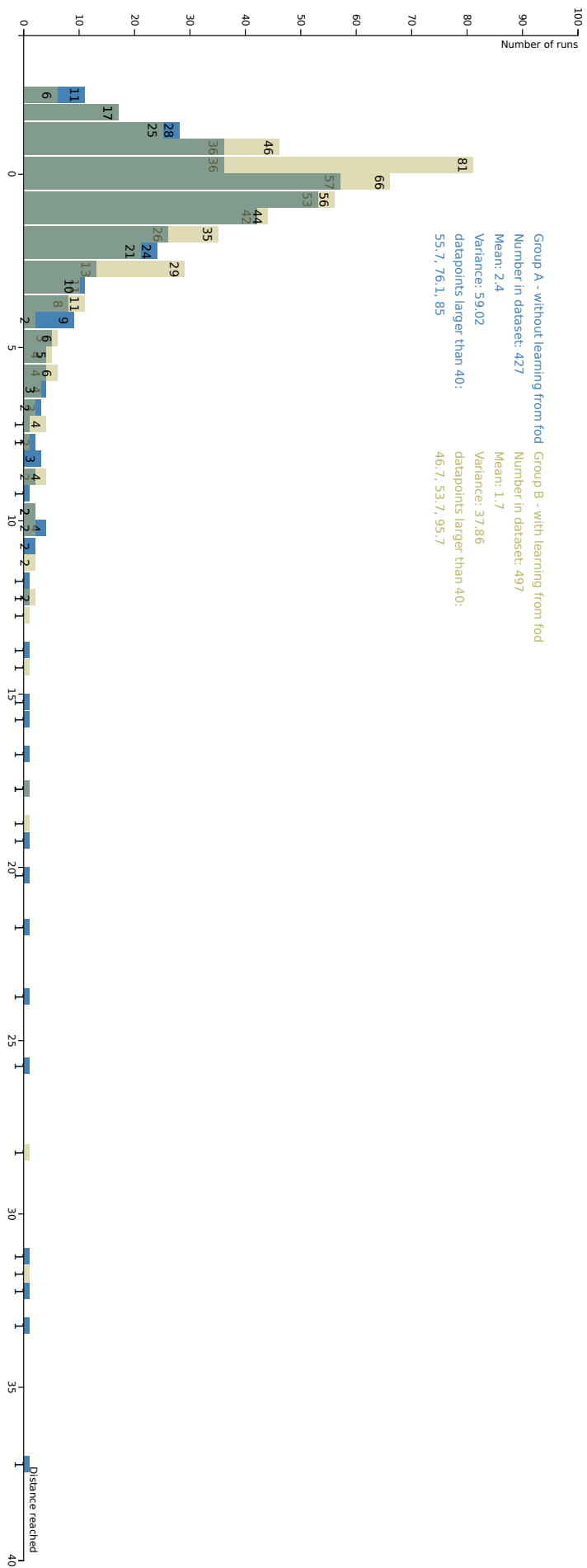


Figure 4.7: Learning bias affecting version gla

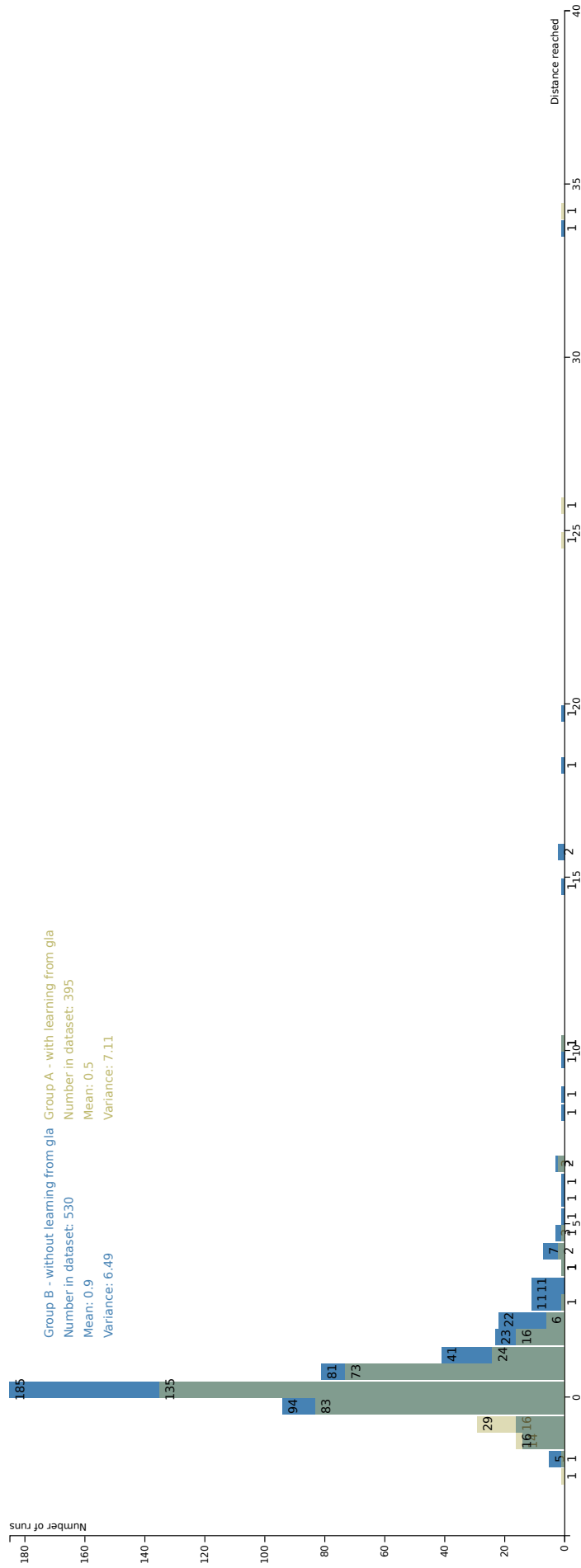


Figure 4.8: Learning bias affecting version fod

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 4.9: 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 4.10: Results of participant choice of version

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

Chapter 5

Conclusion

To conclude, my project has been a success. I have met the success criteria by developing my version of QWOP that is faithful to the original. I have undertaken a controlled experiment with instrumented software that made use of A/B testing to let me control for ordering effects. I have used survey methods to perform both quantitative and qualitative assessment of user perceptions of my software, which has shown that my version improves the experience for most players. Moreover, the user study has shown that players enjoyed my version more as they could reach further, and felt that they had more control over the athlete.

In regards to the language choice, JavaScript was very well suited to the task. Using a dynamically typed language did not create any problems, and as the scope of the project was not a large size, it was clearly a good choice.

Appendix A

Project Proposal

Appendix B

Questionnaire questions

B.1 First Questionnaire

1. What is your age?
2. What is your gender?
3. Have you played QWOP before, and if so, for how long do you think you have played in total?
 - (a) Less than 10 minutes
 - (b) Between 10 and 30 minutes
 - (c) Longer than 30 minutes
 - (d) A long time over multiple sessions
4. On average how long do you spend playing some form of video games during a typical week outside term time?
 - (a) Less than an hour
 - (b) Between 1 and 5 hours
 - (c) Between 5 and 20 hours
 - (d) More than 20 hours

B.2 Second Questionnaire

1. Which version of the game do you think you were more successful at?
 - (a) The first version.
 - (b) The second version.
2. Which version felt like you had more control over the athlete?
 - (a) The first version.

- (b) The second version.
- 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?
- 5. Which version did you enjoy more?
 - (a) The first version.
 - (b) The second version.
- 6. Why do you think this was?