Computer Science Tripos – Part II – Project Proposal

QWOP in JavaScript

George Andersen, gla23, Robinson College

October 19, 2017

Project Supervisor: Dr A. Beresford

Director of Studies: Prof A. Mycroft

Project Overseers: Prof R. Anderson & Prof J. Bacon

Introduction

QWOP is a 2008 ragdoll-based browser video game created by Bennett Foddy. Players control an athlete using only the Q, W, O, and P keys. The aim of the game is to complete a 100m race without falling over, which is a lot harder than it first seems. Once the athlete reaches the end of the 100m, the player is given the time taken to complete the race. If a player can complete the 100m they can then try to do it as quickly as possible. After the game went viral in 2010, people have been trying to break the world record, one speed-run video attempting this on YouTube has 2.5 million views.

QWOP uses a ragdoll physics simulation for the athlete. Each part of the runner's body is a rigid body with its own velocity, rotation and other physical quantities. It has joint constraints with connecting rigid bodies which allows the body to move without ending up in improbable or impossible positions. The simulation also emulates muscles that can be controlled with the Q, W, O and P keys, so that the user can make the runner move. Whilst the Q key is held down, the runner's right thigh is driven forward and his left thigh is driven backward, and the W key does the opposite. The O and P keys do the same but for the calves. Though the objective of QWOP is simple, it has been notorious for being difficult to master due to its controls with the Q, W, O and P keys.

In this project I will make a new build of QWOP in JavaScript using Phaser. Phaser has a physics system called P2 that can be used to recreate the physics behind QWOP.

Phaser has objects called *sprites* which can be used to create the game. Each of these can be attached to a P2 physics body used in Phaser's P2 physics system. I can then build on this system to give these bodies the correct physical properties. More details on how this will work can be found in the Work to be done section.

Starting point

One major codebase the project will be using is Phaser, a JavaScript library designed to make games. So far I have downloaded Phaser and used some of the examples to explore the physics engine.

I have also spent time familiarising myself with the tools I will be using for the project and working out a work flow. These include Git, GitHub, WampServer, Phaser and Sublime text with LaTeXTools.

Resources required

For this project I shall use my own laptop running Windows. My Laptop's specifications: Intel i7 16GB RAM, 1 TB+8 GB SSHD, AMD R9 GPU.

If my laptop fails I can continue development on the MCS machines in the Computer Lab.

I will backup my code and the writeup on a private repository on GitHub.

I require no other special resources.

Work to be done

The project breaks down into the following sub-projects:

Each of these sections include researching how the task can be completed in the Phaser physics engine, and then creating it in the project.

1. Create a ragdoll model of the runner's body out of Phaser sprites with correct distance constraints between sections so that it keeps together whilst collapsing to the floor.

2. Ensure the body has inertia, give the floor friction and apply other physics features so the body acts like a runner rather than collapsing on itself.

3. Make the athlete controllable using the Q, W, O and P keys. Receive this input into the game, and give the correct rotational force to each section of the body as in the original game.

4. Add constraints on how far a joint can rotate. Add these constraints on the model of the athlete so the runner has human-like joint constraints.

5. Add graphics to the body; render an image for each sprite joint in order that together they look like an athlete's body.

6. Calculate and display the distance the runner has moved from the start and how long it has taken.

7. Add collision detection with the floor for the parts of the body that touch the floor when the runner falls over. Restart the game when the athlete falls over.

8. Check whether the runner has finished the race by checking whether the distance travelled is over 100m, and if so, record the time it took to complete the race.

Success criteria

At the end of the project I expect to be able to demonstrate a game similar to that of the original QWOP. 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 will design the study to evaluate the project, giving some thought to ensure that it has ecological validity. In the study I would aim to gain some quantitative data. For example after playing each version for 10 minutes, what was the average distance travelled over these runs for each participant? How much time did each run take? Did the scores of each participant increase over time, and how could these changes be modelled for each version? I would also collect some qualitative data via a questionnaire to discover which game they found most enjoyable, which they thought they did better on, which game felt more responsive to their input, and other comments.

Even though I am not aiming to replicate the behaviour of the original QWOP exactly, another evaluation metric I could use is does the athlete respond in similar ways in both versions to the same inputs? This could be checked by programming the same input into both the original and my version, and checking the athlete goes through the same type of movement. Since I will not have access to the internal workings of the original QWOP game, and therefore cannot code the input directly, I could use an AutoHotkey script to give the desired input consistently.

Possible extensions

If I achieve my main result and still have spare time I shall try an experiment of creating an AI module that attempts to play QWOP. It could be broken down into these sections:

1. Design an encoding scheme (or multiple to see which works best) that can be used to control input to the game. Each instantiated value of the scheme (runner) can be used to control the input to a game. A deterministic version of the game will give the same score each time the same runner plays a game.
2. Provide the AI module with access to the rotations of the different joints whilst the athlete is running so that the AI can use this information.
3. Test runners giving them a fitness score of how far they managed to run before falling over.
4. Make a genetic algorithm that can repeatedly test runners, and mutate them with the effect of increasing the fitness of the runners over time.

Previous work includes this paper [1] that has been written about evolving QWOP gaits. The writer tests various genetic algorithms to evolve a gait (walking motion) for the original QWOP. To do this he uses a similar approach to the one I have described above. The difference is that since he is using the original game, there is no access to the internal

workings of the game, and therefore he can only get information from the game using a screen capture. He uses this to work out the current score, and this is the only information he uses to evolve the runners. Since I am making my own version, I have access to the data in the game, and so I could use this information directly.

I could try reimplementing some of the coding schemes or genetic algorithms the above paper uses, but I will have more options as I have the extra data to use.

The paper found that it was hard to input into the original QWOP so that it acted deterministically. This made it harder for the genetic algorithms to be effective as one input to the game could give different scores on different tries. For this reason I shall make it an extension aim for my version to be deterministic.

Timetable

1. **19th October-1st November** After handing in project proposal, I can start researching how exactly to get the physics engine to do what I want.
2. **2nd - 15th November** Start implementation of the model of runner; Create a ragdoll model of the runner's body out of Phaser sprites with correct distance constraints between sections so that it keeps together whilst collapsing to the floor. Ensure the body has inertia, give the floor friction and apply other physics features so the body acts like a runner rather than collapsing on itself.
3. **16th - 30th November** Make the athlete controllable using the Q, W, O and P keys. Receive this input into the game, and give the correct rotational force to each section of the body as in the original game. Add constraints on how far a joint can rotate. Add these constraints on the model of the athlete so the runner has human-like joint constraints.
4. **1st - 14th December** Start of Michaelmas vacation: Go on student getaway with church, travel home, and visit my sister in Italy with parents.
5. **15th - 23rd December** Add graphics to the body; render an image for each sprite joint in order that together they look like an athlete's body. Calculate and display the distance the runner has moved from the start and how long it has taken.
6. **24th - 28th December** Time off for Christmas
7. **29th December - 8th January** Add collision detection with the floor for the parts of the body that touch the floor when the runner falls over. Restart the game when the athlete falls over. Check whether the runner has finished the race by checking whether the distance travelled is over 100m, and if so, record the time it took to complete the race.
8. **9th - 14th January** Go on Ciccu getaway before moving back to Cambridge for Lent term.
9. **15th - 31st January** Finish off all the sections in work to be done so that I can present the core game working at the overseer review. Write progress report so that it can be handed in before noon on the 2nd February.

10. **1st - 14th February** Contingency time/extensions – extra time here in case previous work sections take longer than expected. If work moves quicker than expected I can start to investigate making the AI or can start designing the user study. Meet success criteria.
11. **15th - 28th February** Work on evaluating the project. Finish designing and perform the user study, and evaluate any extension work. Give time for revisions and improvements of the project.
12. **1st - 14th March** Start writing dissertation main chapters.
13. **15 March - 25th April** Complete the dissertation first draft so that it can be handed in on the 23rd (the day before the first day of Easter full term) so that there is time to incorporate feedback.
14. **26th April - 9th May** Take into account feedback and work on enhancing draft and re-factoring into the final version.
15. **10th - 17th May** Proof read dissertation and then give an early submission so as to concentrate on examination revision.

References

[1] Steven Ray, Vahl Scott Gordon, and Laurent Vaucher. Evolving qwop gaits. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 823–830. ACM, 2014.