Artefact report and commentary

My artefact is a set of computer graphics and physics demonstrations. Written in a variety of languages, namely C++ (using OpenGL) and JavaScript, these have been great fun to research and code as well as providing an opportunity to exercise my skills in project management.

The demonstrations are:

- a wave simulation that is physically accurate (as shown by the accurate simulation of minima and maxima in the double slit experiment), and drawn with 3D lighting effects;
- a bridge simulation game that simulates the physics of rigid and flexible metal links, distorting and breaking under weight;
- a ragdoll game that simulates ragdoll physics and implements collision detection against obstacles that the ragdoll should avoid.

I have also made some experiments during the course of my project:

- a signed distance field test, that takes slices through a 3D scene made up of balls;
- a high quality text renderer, that generates text with varying glow, and rotated using matrices;
- a cube that is made to rotate using matrix operations, and has diffuse lighting.

I have done some programming before, but mainly in the context of problem solving and algorithms. However, I have always been interested in the graphical side. Demoscene productions have always awed and inspired me, and I have always wanted to do some graphics programming.

Planning and organisation

I had anticipated that learning OpenGL would be a massive endeavour. This is why I decided to start my project as soon as possible, planning small sub-projects to help me to get to grips with it.

Right from the beginning before I had even started writing any code, I carefully planned my program, structuring it into three parts. The most important of the parts, the “library,” would be a set of useful and reusable code, such as mesh management, texture and lighting code, that I could use in each of my sub-projects without having redundant copy-and-pasted code everywhere.

Before I started coding anything, I would research the sources describing the technical aspects thoroughly. For instance, my physics simulation would not have been so successful if it weren't for the many Verlet integration sources and information that I had found before I had started writing the physics engine.

I planned the intermediate experiments so that the skills I gained from coding them would transfer into my final demonstrations. For example, the cube example gave me the requisite knowledge about diffuse lighting to implement part of Blinn-Phong shading in my final wave demonstration, and it also taught me vector dot products, which I later used in my bridge demonstration too.

I knew from past experience that time management would be an issue for me when doing the project. I worked very hard to combat this from the beginning of the project; by setting myself targets and deadlines on my Timeline I could keep on track.

I used GitHub, an online service designed for storing programming projects. I could put my code on this website and it would be automatically saved along with all its previous revisions. It meant that I could roll back changes that didn't work. I also put my project documentation on this website, as it meant that I could access it both from college and at home, without having to carry around memory sticks.

I put deadlines on the calendar on my phone. It would remind me a period before the deadline, so

that I could prioritise things as they came. Often I would find that I had to juggle two or three time-consuming things on top of my EPQ, such as UCAS applications or physics coursework, and it was at these times that my calendar app came to my rescue, as I could efficiently see which deadline was coming up first and prioritise accordingly.

In addition to this, nearer the end of my project, I tried something new, which turned out to be surprisingly effective. I would promise my friends that I would get a certain part of my project done by a certain date, at which time I would show them my progress so far. Closer to the date my friends would remind me to get on with the project, with the expectation that they could get to play around with my demonstrations once I had finished certain parts of it. Psychologically I found this extremely effective as I felt a strong pressure not to disappoint my friends' high hopes.

Research and development

My research encompassed a large range of sources. As my artefact is mostly computer-based, most of the relevant information that I have found has been on the internet. However, this web-based information has taken the forms of blog posts, research papers. Videos of presentations given at conferences, and their corresponding slides, have also come in very useful.

My sources have mostly been factual or outlined procedures on how to achieve things using different programming techniques. They have mostly been independent of each other, however where there has been overlap they have been in agreement, and often they have even cross-referenced each other. For example, the Valve paper on text rendering was cross-referenced from Stack Overflow, and several other papers on OpenGL textures.

I think over the course of the EPQ I have greatly improved my abilities in not only finding sources, but also using them effectively. I have become better transforming the information in my sources into something that is useful to my project. Most of the time this involves writing the code that implements the theoretical concepts that I have read about, but sometimes this also includes working things out mathematically and drawing diagrams to help understand things.

At the beginning of the project, I struggled to implement mesh rendering because I did not know how to effectively turn the theory into something I could understand easily. In addition, I made two errors: I did not write good test code that exercised mesh rendering, and I did not realise that meshes were not the easiest solution to the problem that I was having, and that there were much better approaches. Once I identified these problems, I took action in remedying them -- I changed the wave simulation from being mesh-based to being heightmap based, and I started setting myself smaller targets to achieve.

The physics engine, on the other hand, worked fantastically from the beginning. This is because I started work on the physics engine relatively early, in April, and continued researching and working on the engine over the course of four months. In addition an important factor contributing to the success of the project was the fact that I set myself checkpoints after implementing certain features of the physics engine, which I would test by writing new JavaScript experiments around. For example the ribbon example was written after I had implemented length constraints; and the ragdoll example was written after collision detection was coded into the engine.

The final demonstration, the bridge simulation, was intended to be slightly larger in scope. Based on the experience and research I had done for previous experiments, I was able to use Verlet integration and length and angle constraints, which worked because of the thorough research I had done beforehand. However, I could not write the collision detection in time. I had expected to be able to use the code I had written to detect collisions in my ragdoll game. This was not the case as my algorithm detected collisions between a line and a point, whereas in my bridge simulation I had to find collisions between two lines -- this was an oversight on my part. I did not have time at the end to research and implement this.

In my project there was unfortunately not much scope for primary research. I attempted to reach out to two well-known programmers by email for information, however I did not receive a response from either of them.

My friends, however, were much more useful in giving primary feedback. There was a constant stream of verbal support and criticism from them during the course of my project, and at the end when my bridge simulation was complete many of my friends tried it out and gave me feedback, both verbally and online. This feedback was very useful to me as it kept me on the right track, and allowed me to make improvements to the simulation that I otherwise would not have spotted.

Skills developed

I learnt that it is important to start the project early. Leaving things to the last minute doesn't work in a project as big as this one. In addition, I learnt to set small quantifiable targets and checkpoints and testing those targets by creating experiments in order to test that these work. I did not do this effectively in the first stages of my project, and this led to difficulties.

If I were to do my project again, or a similar project, I would definitely carry on setting myself quantifiable targets. In addition to being a great way of measuring progress, they are very good at motivating and guiding future work. When I set achievable targets that I could see the results of. stopping me from going off track and encouraging me to continue.

I learnt to use many technical tools. OpenGL is an industry-standard graphics library that allows efficient rendering of 2D and 3D.

Git is a version control system, designed for software projects. It is used professionally in open-source projects, such as the Linux kernel. It makes it possible to write code and have previous revisions saved. It also supports branching, stashing, and merging, which allowed me to work on two different features at the same time on separate branches of the code, and merge them into one at the end once I've finished. Git is notorious for being unfriendly towards beginners, but now I feel very comfortable managing Git repositories from the command line.

GitHub makes Git available online. It means that I can access my code from anywhere, and in the unlikely case that my laptop dies, I have not lost my code as it is backed up in the cloud. GitHub is the most popular online code repository service.

I have greatly enhanced my skills in writing code in C++. Previously I was unfamiliar with the new features of C++ introduced in the latest revision of the language, C++11. This project has helped me to get to grips with these features, whilst increasing my confidence in using the various, more obscure, parts of the language.

HTML5 and canvas is the latest trend in web development, and as part of my EPQ I managed to get to play with this new technology. It forms the heart of modern web apps, and my experience with it making the various physics demonstrations has allowed me to get very familiar with it.

If I go into the software development industry, then the technical skills that I have acquired would definitely benefit me greatly -- especially as I am not studying Computing at A-level. However, I have also gained more general skills that can be applied to any situation.

My project and time management skills have improved dramatically, and I have discovered new ways of tackling time issues, as outlined above. I felt that I was much more in control by the end of the project, as I was confident that I would be able to deliver by the deadline that I had set myself.

I am now much better at reading scientific papers and picking out the relevant parts from them. Before I would be intimidated by strange language or unfamiliar symbols. However now I know where to look for the relevant parts, and also where to to look up what unfamiliar mathematical terminology means.

In addition I feel like I am much stronger at problem solving now. When there was a problem in my

code (in fact, this was the situation most of the time), I had to track down the source of the bug -- sometimes it was a careless mistake while coding, but sometimes the problem turned out to lie beyond the realms of the code, and I would discover that I was actually using the wrong algorithm for the task. My intuition for finding and correcting mistakes improved through the course of the project, and my perseverance also improved.

Overall, the Extended Project has definitely made me grow as a person, and I feel much better equipped to tackle larger real-life problems using the toolkit that I have built up over the course of the Project.