Oliver Goldstein: SPE Report 2016
og14775

# Software Product Engineering Individual Report

## Introduction

I developed the project idea, consisting of the interface and game idea. I was responsible for email and phone communication with Anna, the Creative Director of @Bristol. I co-ordinated the stock list with Richard Grafton. I created an initial code base in C and constructed the initial Arduino circuit setup linking capacitive fruit to directional controllers through serial ports to a 3D Unity game level, with enemies. I constructed the (1m x 0.75m x 0.5m) physical wooden housing for the apples using AutoCAD, according to specification. In game, using a variant of C#, I programmed dynamics of game camera movement, to allow gentle acceleration, avoiding motion sickness, a major problem of HMD's and implemented asynchronous time warping. I implemented, the art, SFX, and physics of asteroids; the major enemy within the game - in a pair. I produced in game sounds on Logic Pro. I created Health & Safety spec & developed a shared Google Drive doc to document the iterations, performed. I co-ordinated the overwhelming majority (15) team meetings over the Facebook Group/Google Drive we setup. For the technical setup, see Fig. 1 and game setup - Fig. 0. The video (which I created): https://www.youtube.com/watch?v=FXV-qnR0jFg

Fig 0. Basic Technical Game Overview

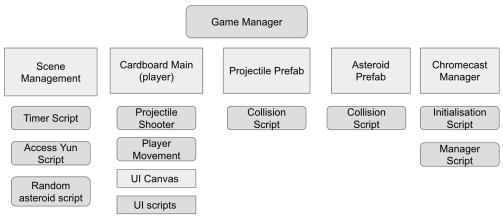
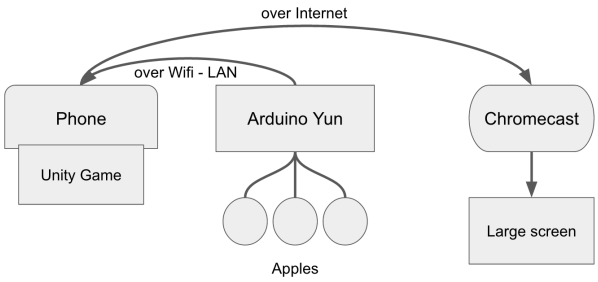


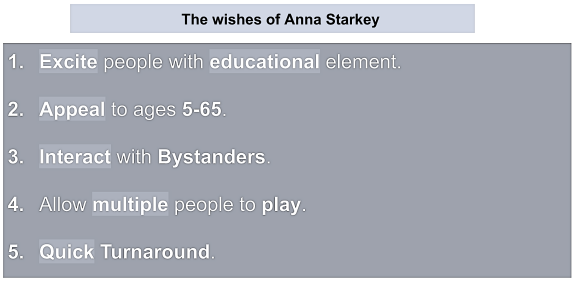

Fig 1.




## Early Stages

Project structure involved using Agile, Lean & XP (Extreme Programming) methodologies, iterating towards an MVP which we demonstrated to Anna using her feedback to guide development, placing priority on technologically risky features early on, doing integration tests early on - a methodology from continuous integration. We faced an open-ended brief. Therefore, it was crucial that our ideas aligned with desires of @Bristol. We met with @Bristol, discussed our idea and agreed on a set of principles. See Fig 2.

Fig 2.

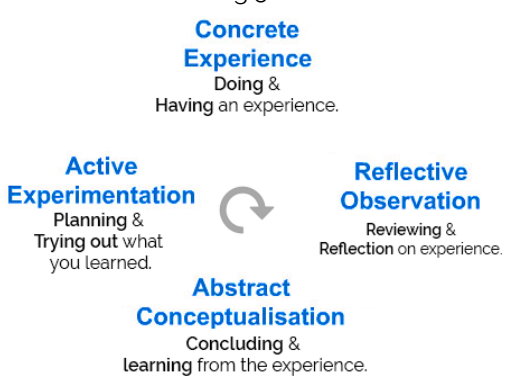



Three things I kept in mind from the beginning, first; avoid quarrelling with team members, second; do this by compromising when needed. This was done, to my knowledge, successfully. Intuitively I felt I lead the team, though vocalising this I felt was unnecessary. I found balance between refraining from bossiness, yet still pushing things forward. This informal hierarchy (flat management structure) promoted a healthy and argument free process. Thirdly, listening to the talk by Sam Phipps, I resonated with the idea of making each weekly task as small as possible, making the project goals achievable. The project was holistic and we tried to loosely couple it, for the sake of simplicity with each person acting as a holon (both a part of the project and a modularised whole). The project used skills in and outside of CS & I plugged many gaps in the project, often flitting between different project parts.

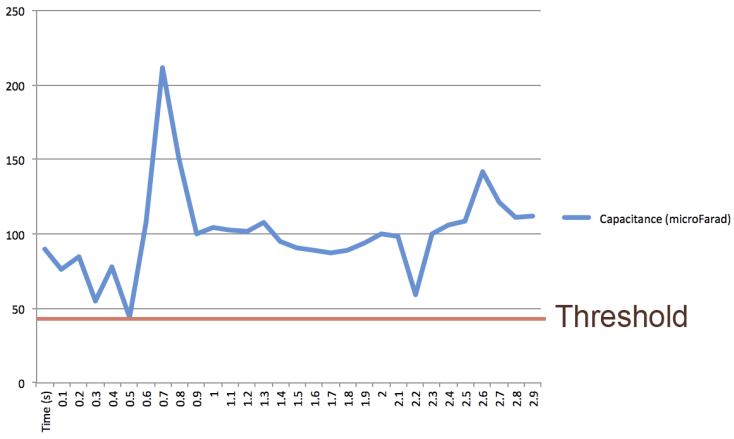
## Structural Quality: A Case Study Iteration

I used the Kolb & Fry learning cycle to reason about iterations (Fig 3), starting with a concrete experience. I started the code base using the Arduino, creating a prototype demonstration of the initial concept. It consisted of four capacitance sensing apples, linked via serial ports to a level map I hand crafted in Unity - a game with basic enemies. We then tested gameplay at this point and decided a variable force dependent on distance between hand and apple would be more fun and challenging. Here, I faced a challenge in accuracy. I wanted a ratio of distances from hands to each apple to and to then output a corresponding (x,y) value. This was the Concrete Experience. The poor quality of our sensors meant that the values rapidly and unpredictably fluctuated, meaning the ratios fluctuated as well, creating jagged & poor gameplay. This can be seen in Fig 4, which shows capacitive change over 3 seconds. Specifically the user keeping his hand at a constant distance from the apple. The distance remained the same but the capacitive values changed wildly. This was the process of Reflection & Observation. I thought about a simple solution to the problem and I

Fig 3.



realised that often it's less about code elegance and more about getting the job done (KISS) This was the abstract realisation or Abstract Conceptualisation. I crudely thresholded (as seen in Fig 4.) each capacitive value beyond a certain value, mapping it to an integer which I added in a loop for each sensor, with each value mapping to a direction. However, this method sacrificed resolution as gauging distance from a users hand was binarized rather than "analogue", in return for consistent gameplay. Keeping in mind frame-rate and its contribution to motion sickness, I used an observer design pattern to deal with data - the Arduino notifies the computer upon a change of hand state, allowing for a smoother, more robust and efficient gameplay with about a 80% reduction in game data pushes made from 200 per minute to less than 40. This represented an Active Experimentation of the principles from the abstract conceptualisation. I used clean commenting and modularised code to allow the code to be later refactored easily.




## Understanding & Addressing Users Need: A Case Study

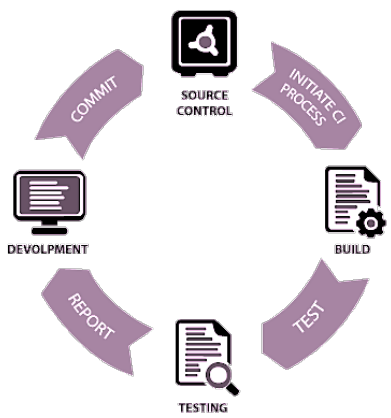
The user's need was given by the demographic of @Bristol i.e. a diverse range of people in terms of age, gender etc. As such, Anna spoke about her desire to prevent users, especially younger schoolchildren & the elderly experiencing nausea from our project. Motion sickness was a problem I therefore focussed on, especially after tinkering with Cardboard and feeling intense long lasting nausea. I sourced inspiration from the Developers of Oculus Rift (see right). A feature I programmed was camera movement. Acceleration, I researched is a primary determinant of vestibular imbalance in VR. I therefore scripted gentle (linear) acceleration for all game camera movements. Frame rate & latency were also determinants, so Arthur & I chose asteroid models with a low number of polygons to reduce rendering time. In addition, I refactored the game loop structure to implement basic asynchronous time warping - a concurrent technique, pioneered by Oculus, which stipulates that if the scene update method is delayed the scene draw method still has the previous scene of the landscape and so upon head movement it can still render the previous scene, except for changing input from extraneous interfaces which may have latency requiring the update scene method e.g. shooting. This massively reduced the frame rate in periods of high latency making for a smoother gameplay, reducing nausea.

## Linking it all together: A tale of Team Management, Integration & Deployment

We modelled development on continuous integration (CI), see Fig 5. We started a master branch in Git, each one of us forking from this creating individual branches. We ensured regular builds, and whenever anybody did build and commit I oversaw smoke testing of essential functionality using physical equipment from the Merchant Venturers Building (MVB). Eventually we physically centralised hardware in the MVB, streamlining testing. I kept a log of previous bugs I found and made sure to test those (a form of regression testing). Merging was not done regularly due to the configuration files of Unity creating difficulties. We ended up manually merging branches i.e. copying files over.

Automating deployment was challenging, considering the interfacing of software and hardware, so instead of continuous (automatic) deployment we used continuous delivery, where at any point we could manually deploy to production.

Fig 5.




I made small unit tests for minor game logic bugs that we experienced, but not for rendering bugs which were not easily tested programmatically. Integration testing was done early on and regularly (see early stages). Acceptance testing (see Fig 6), was performed with canary tests, testing on a small set of users in the MVB, receiving feedback and sometimes iterating on a game feature. We also used split tests (A/B testing) on the skybox designs due to a minor conflict between myself & the team on whether to use a skybox planet earth or an actual 3D model, so we ran both in parallel asking passers by in the MVB to rate the aesthetic of the game design. We found passers by preferred the skybox and the 3D model increased the frame rate so decided against the model version. Managing work with varying skill sets and egos was challenging. Some team members wanted a loosely coupled section they could own, which would have

left others out. On the majority of occasions we used pair programming, continuously reviewing each others code, giving feedback as we worked, hoping the social aspect would promote work ethic.

Fig 6.