Technical Design Document(TDD)

version 0.3

Table of Content

1. Introduction

2. Architecture

3. Coding Standards

4. Tools

5. Game Engine

6. Risks and Contingencies

7. Security

8. Revision Control

9. Physics

10. Input/Output (I/O)

11. Hardware Consideration

12. Multiplayer and Internet

13. Graphics

14. Sound

15. Localization

16. Research and Development (R&D)

17. Prototype

18. Final Prototype

1. Introduction

Our project idea is to create a Haptic feedback system which will allow the user to experience physical feedback when playing either a Video game or simulation. We plan to use a modular system that will allow for multiple points of contact on the user with the aim of providing an immersive gaming experience.

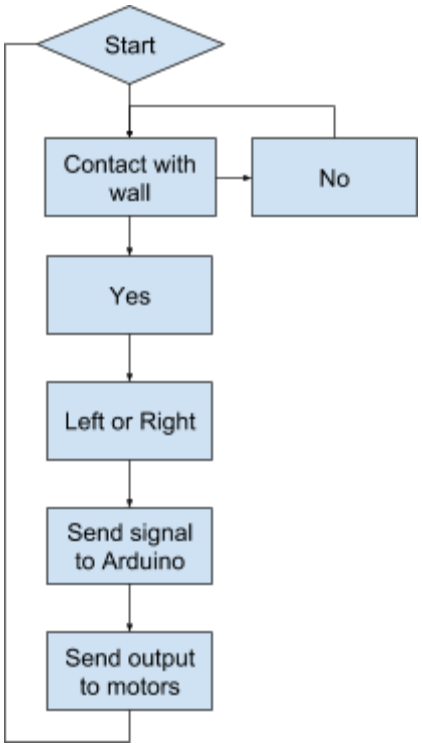
The main focus of this project is the hardware, as of which we are creating a rapid prototype of a Haptic feedback system. We've set out with a goal of providing a product which has the capabilities of providing a physical form of feedback, that will allow the user to feel more immersed in a game/simulation of some form. A large part of our project has been comparing different modules that we have made along the way, as well testing different motors to find what the best configuration is, in order to achieve our goal.

Along with our hardware, we've also developed a simple maze game to demonstrate the products features, in addition to this, we also have written sample software which can demonstrate the feedback in a controlled scenario.

2. Architecture

Unity>>Arduino>>Motor Feedback>> User

We used unity to create a short game that allows the user to navigate around a maze without the use of their sight. The game comprises of a series of walls which forms a maze with additional obstacles to navigate around. The player has two hit boxes for each side of their body. When they interact with the walls (which also have a hit box) Unity sends a signal to the Arduino which will then fire the appropriate motors.



3. Coding Standards

For the project we have used C# for the Unity scripting and game controls, and arduino's implementation of C / C++ for the Arduino coding.

We have decided to use common coding standards such as camel case for variable names, and have limited the use of global variables to only be used when necessary

4. Tools

3D asset creation has been performed in Blender, with textures being created and/or modified in Adobe Photoshop/Illustrator CS6 / Adobe Photoshop CC.

Sound files (if implemented/required) will be recorded and/or edited in Audacity.

Unity has been used to create the game / experience environment
Arduino desktop application to write and compile the code to operate the hardware
3D Printing will be done with either Blender or Autodesk fusion 360

5. Game Engine

Our project will be created using Unity 5, due to familiarity with the software, alongside the ability to easily communicate and send data to an Arduino, allowing us to control the physical device outlined in this project.

6. Risks and Contingencies

One risk is that we are using relatively cheap hardware, In order to ensure that we don't get caught out by a broken motor we have made sure to order a surplus of motors to ensure that we have spares
Errors with the game with exporting data from unity. Incase our game crashes, we have written some basic code which will fire the motors to provide the user with a simulation of the game. Just incase our game doesn't work on the day for a useability test we can still gather data for the comparison of two different types of motors. This will allow us to improve our project for the next version

New versions of code not working: To avoid this, we have used GitHub to publish our code, this allows us to use their version control which mean that we will always have a backup of previously working code if we manage to break it beyond repair.

7. Security

Our project is neither connected to the internet or handles personal Data, therefore Security isn't a major concern for our project.

8. Revision Control

Our project will make use of a GitHub repository for our version control, using the "master" branch for each "release" (E.g. pre-alpha, alpha, beta etc.) of the project. It is intended that good Git practice will be used throughout the project to aid development. Alongside a GitHub repository, two backups will constantly be created/updated to ensure that a copy of each revision (both builds and development files) will remain available. One backup will be stored locally (via USB / HDD Storage) whilst another copy will remain cloud-based.

9. Physics

The game collision physics between the player model and the walls of maze and the player model is also under the effects of a gravitational force.

10. Input/Output (I/O)

The game will output data to over the USB serial port to an Arduino which will then be used to drive several motors, in order to provide Haptic feedback. The game outputs data to the Arduino based on a directional hitbox system that allows the user to experience feedback on either their left or right side of their body. The input for the game will be using the WASD keys to navigate around the map and the mouse input to start the game. In future versions of our project we hope to use a leap motion device which will map the hands allowing the user to interact with the environment more freely. In addition we would like to make the map 1-1 scale in order for the user to experience a more immersive form of locomotion.

11. Hardware Consideration

Our goal is to make use of Virtual Reality (run via a Desktop PC, rather than mobile devices such as a phone placed in a Gear VR or Google Cardboard headset). The application itself should not prove resource intensive beyond that of HTC Vive/Oculus minimum specification.

For reference, these are:

HTC Vive[1]		Oculus Rift[2]	
CPU	Intel i5-4590 / AMD FX8350	CPU	Intel i3-6100 / AMD Ryzen 3 1200, FX4350
GPU	NVIDIA GeForce GTX 970 / AMD Radeon R9 290	GPU	NVIDIA GeForce GTX 960 / 1050 Ti / AMD Radeon R9 290
RAM	4GB or more	RAM	8GB or more
OS	Windows 7 SP1, Windows 8.1 or later, Windows 10	OS	Windows 10 or newer

12. Multiplayer and Internet

[Describes the multiplayer experience including the communications protocol and the online presence if any]

Currently there are no plans for multiplayer and/or networked features.

13. Graphics

[Describes all the graphics objects in the game and the 3D pipeline and the user interface]

The project's visual style is initially hopeful of photorealism, as creating a strong sense of immersion is a fundamental part of the project. A range of environments will be created to provide lifelike simulations / experiences.

When it comes to the project's user interface(s), the goal is to remain as minimal as possible. The project will include navigation menus and pause menus, but we aim to display no additional user interfaces during the course of a experience/scenario.

For our latest advancement with making a game rather than a simulation we have gone for a more minimalistic approach to things.

[1] https://www.vive.com/uk/support/vive/category_howto/what-are-the-system-requirements.html
[2] https://support.oculus.com/170128916778795/

14. Sound

Our initial idea was that sounds will be that of a natural environment from scene to scene, as our goal is aimed more towards creating a strong sense of immersion. Sound files will most likely come from a range of sources; including online sources and real world recordings gathered by ourselves.These will be implemented in Unity within a 3D environment to provide spatial sound where appropriate.

Due to a change in direction early on in the project we moved away from the idea of creating a simulation environment and instead created a maze style game, which does not use sound as part of the experience.

15. Localization

Currently, we're aiming our product at the UK only to the nature of the project involving the creation of a physical device. Our software is therefore only provided in English, being the most common language within the UK. The software has not taken into account the need for localization, therefore would require some adjustments if the target territories change.

16. Quality Assurance(QA)

When making the hardware element of the project we wrote several different versions of arduino to test the motors at different stages of development to ensure that they were all wired up correctly. This can been seen on our GitHub repo there are many different versions of the code.

For each build of the game we tested it with the arduino code to see if the game and the hardware would run together.

17. Prototype

Proof of concept Prototype:
To test the basic design elements we created a basic prototype but using a wrist support with 3 small rumble motors attached. We used an Arduino Uno to drive the motors, which we plan to use in the future to connect the motors to a simulation using Unity.

Alpha Build for Usability testing:
Through prototyping our ideas we have found that we need to provide a variety of different sensations through our feedback system. Some of these effects are listed as followed:

- Light feedback for brushing a surface
- Medium feedback for impacting a surface

- Heavy for continuous contact with a surface

We have also found from usability testing that people prefer the feel of the rumble motors but also like quick response time of the coin cell motors. Going forwards from this we have decided that our future design will implement both types of motors on both the left and right arm modules. This decision will allow us to provide the best of both of the previous designs, the coin cell motors would be good to provide a light feedback with short bursts. Then for the medium feedback we could have the coin cell motors fire for longer bursts and then finally for the Heavy feedback we could have the rumble motors be used in combination with the coin cell motors.

From the testing we found that our modular system worked quite well and was able to fit every user, whereas our proof of concept design had issues with being adaptable. The current design is compact and can be transported with ease. However we do need to create a more professional arduino housing and a better way to manage cables. In addition this problem could be solved with the use of a bluetooth module which would allow us to communicate with each module wirelessly rather than running a wire to each one. This will especially be important to implement if we make create modules for the legs.

18. Final Prototype:

After our testing of the Alpha build, we've made a couple of changes to our design. We've created two new modules which contain three motors each. Two being coin cell and one being rumble. The three different variation of feedback is slightly different to what we set out to do after the Alpha build. The three types are now: Rumble on their own, Coin cell on their own, and both Rumble and Coin cell together. We felt that this would give a better variation of feedback then our previous plans.
We have also improved our main rig by adding a fabric wrapping around the wires that go to each module. This has given our product a sleeker look as well as preventing cables getting tangled. One thing that we didn't improve was the arduino housing, this was mainly because we needed to keep the board easily accessible due to complications with wires breaking. Without a better solution for connecting the motors to the arduino we didn't want to cover the arduino.

Circuit Details:

For the final build of the project we have used two rumble motors and four coin cell vibration motors separated into two modules (left and right).

The left modules circuitry is as follows

Coin cell vibration motor 1 is connected to the Digital PWM pin 3(PD3), and to ground
Coin cell vibration motor 2 is connected to the Digital PWM pin 5(PD5), and to ground
Rumble Motor is plugged into Digital PWM pin 4(PD4), and to ground

The right modules circuitry is as follows
Coin cell vibration motor 1 and 2 are both connected to Digital PWM pin 7(PD7), and to ground
Rumble Motor is connected Digital PWM pin 6(PD6), and to ground

For this version of the project we have also created two separate builds of the maze game, so it can be run with and without the use of VR. Both games are work in the same way as before, the player has to navigate a maze in total darkness using the directional feedback from the modules to find their way around. The standard game will use WASD to control the player model, where as the VR port allowed the user to move around using the VR controller.