Abstract

The following document details requirements set for the development of software for the DreamZBox 2.0 game console by clients Megan McCormick and Corinna Brown. It further explains how the CS29 Dreamz Catcherz planned to meet these requirements by developing a game and a dashboard capable of launching it. The CS team is also required to link the console’s controllers to it via Bluetooth. Each team member’s individual efforts and personal experiences during the project are also detailed. Each team member has positive experience creating the top-down two dimensional maze game, Maze of Dreamz.
CONTENTS

1 Introduction to Project ........................................ 8
   1.1 Combined Discipline ........................................... 8
   1.2 Client .......................................................... 8
   1.3 Members and Roles .............................................. 9
   1.4 Current Events and Outcomes ................................. 9
   1.5 Contributing to the Project .................................. 9

2 Requirements Document ........................................ 10
   2.1 Introduction ..................................................... 10
      2.1.1 Purpose ..................................................... 10
      2.1.2 Scope ....................................................... 10
      2.1.3 System Overview .......................................... 10
      2.1.4 Definitions ................................................ 11
   2.2 System Requirements ......................................... 12
      2.2.1 Functional Requirements ................................. 12
      2.2.2 Usability Requirements .................................. 12
      2.2.3 Performance Requirements ............................... 13
      2.2.4 System Interface .......................................... 13
      2.2.5 System Modes ............................................. 13
      2.2.6 Information Management ................................. 13
      2.2.7 Packaging .................................................. 14
   2.3 Verification and Metrics ...................................... 14
      2.3.1 Functional Requirements ................................. 14
      2.3.2 Usability Requirements .................................. 14
      2.3.3 Performance Requirements ............................... 14
      2.3.4 System Interface .......................................... 14
      2.3.5 System Modes ............................................. 15
      2.3.6 Information Management ................................. 15
      2.3.7 Packaging .................................................. 15
   2.4 Closing Statement .............................................. 15
   2.5 Changes in response to COVID-19 ......................... 15
      2.5.1 New barriers to completing the project ............... 15
      2.5.2 Plans to overcome these barriers ....................... 16
      2.5.3 Potential issues for delivering outcomes ............. 16
   2.6 Changes in response to Client Meetings .................. 16
   2.7 Appendix ....................................................... 16
   2.8 Gantt Chart ..................................................... 17
3 Design Document

3.1 Introduction ................................................................. 18
3.2 Scope ............................................................................. 18
3.3 Context ............................................................................ 18
3.4 Design Languages ............................................................ 18
  3.4.1 UML Case Diagram ..................................................... 18
  3.4.2 State Transition Diagram ............................................. 19
  3.4.3 UML Class Diagram ..................................................... 19
  3.4.4 Sequence Diagram ...................................................... 19
  3.4.5 UML Component Diagram ......................................... 19
3.5 Views .............................................................................. 19
3.6 Summary .......................................................................... 19
3.7 Glossary .......................................................................... 20
3.8 Design Stakeholders .......................................................... 20
3.9 Concerns ........................................................................... 20
  3.9.1 Fun Experience ........................................................... 20
  3.9.2 Functional Interface .................................................... 20
  3.9.3 Descriptive Interface ................................................... 20
  3.9.4 Multiplayer Game ........................................................ 21
3.10 Hardware to Software Interface ........................................... 21
  3.10.1 Design Elements ........................................................ 21
  3.10.2 Design Language – UML Case Diagram ....................... 22
  3.10.3 Design Mentality ........................................................ 22
3.11 Dream Modes ................................................................. 23
  3.11.1 Design Elements ........................................................ 23
  3.11.2 Design Language – State Transition Diagram ............... 23
  3.11.3 Design Analysis ........................................................ 23
3.12 Dashboard ........................................................................ 23
  3.12.1 Design Elements ........................................................ 24
  3.12.2 Design Language – UML Component Diagram ............. 24
  3.12.3 Design Language – Sequence Diagram ......................... 25
  3.12.4 Design Analysis ........................................................ 25
3.13 Game Experience ............................................................. 26
  3.13.1 Design Elements ........................................................ 26
  3.13.2 Design Language – Use Case Diagram ......................... 27
  3.13.3 Design Analysis ........................................................ 27
3.14 Game Logistics ............................................................... 27
  3.14.1 Design Elements ........................................................ 27
  3.14.2 Design Language – UML Class Diagram ....................... 28
  3.14.3 Design Analysis ........................................................ 28
4 Tech Review – Tristan Hilbert

4.1 Introduction ...................................................... 30
4.2 Sound Data Management ........................................... 31
  4.2.1 Criteria ......................................................... 31
  4.2.2 Choices ......................................................... 31
  4.2.3 Discussion ...................................................... 32
  4.2.4 Conclusion ...................................................... 32
4.3 Software Design Pattern ............................................ 32
  4.3.1 Criteria ......................................................... 32
  4.3.2 Choices ......................................................... 32
  4.3.3 Discussion ...................................................... 33
  4.3.4 Conclusion ...................................................... 33
4.4 Software Multi-threading Management ............................... 33
  4.4.1 Choices ......................................................... 34
  4.4.2 Discussion ...................................................... 34
  4.4.3 Conclusion ...................................................... 34
4.5 Closing Remarks ................................................... 35

5 Tech Review – Cory Hayes

5.1 Introduction ...................................................... 35
5.2 Art Assets ......................................................... 35
  5.2.1 opengameart.org ............................................. 35
  5.2.2 craftpix.net .................................................... 35
  5.2.3 itch.io ......................................................... 35
  5.2.4 Review and Selection ......................................... 36
5.3 User Interface Frameworks ........................................... 36
  5.3.1 LiveCode ....................................................... 36
  5.3.2 Qt ............................................................. 36
  5.3.3 Kivy .......................................................... 36
  5.3.4 Review and Selection ......................................... 36
5.4 Programming Languages ............................................ 36
  5.4.1 C++ .......................................................... 37
  5.4.2 Python ........................................................ 37
5.4.3 Java ................................................................. 37
5.4.4 Review and Selection ............................................ 37
5.5 Conclusion ............................................................. 37

6 Tech Review – Maxwell Evdemon 37
6.1 Introduction ........................................................... 37
6.2 USART Connections ................................................. 38
  6.2.1 USART Introduction ............................................. 38
  6.2.2 USART Options ................................................ 38
  6.2.3 USART Option Comparison .................................... 38
  6.2.4 USART Conclusion ............................................. 39
6.3 I2C ................................................................. 39
  6.3.1 I2C Introduction ................................................ 39
  6.3.2 I2C Options .................................................... 39
  6.3.3 I2C Option Comparison ........................................ 40
  6.3.4 I2C Conclusion ............................................... 40
6.4 HDMI Functionality ................................................ 40
  6.4.1 HDMI Introduction ............................................. 40
  6.4.2 HDMI Options ............................................... 40
  6.4.3 HDMI Option Comparison .................................... 41
  6.4.4 HDMI Conclusion ............................................. 41
6.5 Conclusion ............................................................. 42

7 Tech Review – Haofeng Tian 42
7.1 Introduction ........................................................... 42
7.2 METRICS ............................................................. 42
7.3 TECHNOLOGY ....................................................... 42
  7.3.1 Music generation ............................................... 42
  7.3.2 Game engine .................................................... 43
  7.3.3 Image loading .................................................. 43
7.4 CONCLUSION ........................................................ 44
  7.4.1 Game engine .................................................... 44
  7.4.2 Music generation ............................................... 44
  7.4.3 Image loading .................................................. 44

8 Weekly Blog Posts – Tristan Hilbert 44

9 Weekly Blog Posts – Haofeng Tian 60

10 Weekly Blog Posts – Maxwell Evdemon 66

11 Weekly Blog Posts – Cory Hayes 85
13 Project Documentation

13.1 Maze of Dreamz
13.1.1 Running The Software
13.1.2 Architecture Description
13.1.3 Gameplay

13.2 DreamZDash
13.2.1 Installation
13.2.2 Using The Dashboard
13.2.3 Architecture

13.3 Bluetooth
13.3.1 Installation
13.3.2 Using the Bluetooth Connection
13.3.3 Bluetooth Connection Design

14 Recommended Technical Resources for Learning More

14.1 Maze of Dreamz
14.2 DreamZDash
14.3 Bluetooth

15 Conclusions and Reflections

15.1 Haofeng’s Reflection
15.1.1 What technical information did you learn?
15.1.2 What non-technical information did you learn?
15.1.3 What have you learned about project work?
15.1.4 What have you learned about project management?
15.1.5 What have you learned about working in teams?
15.1.6 If you could do it all over, what would you do differently?

15.2 Maxwell’s reflection
15.2.1 What technical information did you learn?
15.2.2 What non-technical information did you learn?
15.2.3 What have you learned about project work?
15.2.4 What have you learned about project management?
15.2.5 What have you learned about working in teams?
15.2.6 If you could do it all over, what would you do differently?

15.3 Cory’s reflection
15.3.1 What technical information did you learn?
15.3.2 What non-technical information did you learn?
15.3.3 What have you learned about project work?
15.3.4 What have you learned about project management?
15.3.5 What have you learned about working in teams? .................................................. 95
15.3.6 If you could do it all over, what would you do differently? ................................... 95

15.4 Tristan’s Reflection ................................................................................................. 95
15.4.1 What technical information did you learn? ......................................................... 95
15.4.2 What non-technical information did you learn? .................................................. 96
15.4.3 What have you learned about project work? ........................................................ 96
15.4.4 What have you learned about project management? .......................................... 96
15.4.5 What have you learned about working in teams? ............................................... 96
15.4.6 If you could do it all over, what would you do differently? ............................... 96

References .................................................................................................................. 97

16 Appendix 1 ............................................................................................................... 97
17 Appendix 2 ............................................................................................................... 98
18 Appendix 3 ............................................................................................................... 103

List of Figures
1. The DreamZBox 2.0 Gantt chart displays the plan for the next several months in implementing the viewpoints. The Gantt chart also includes other tasks which may take precedence during development. The Gantt Chart establishes the latest possible deadline for the given pieces of the project. ...................................................... 17
2. The UML Case Diagram establishing the Inputs and Outputs of the system. It’s important that the external actors of this diagram could be translated to components in the DreamZBox 2.0 hardware. Also, the different arrows establish relationships between the provided services. The “extends” arrows provide an extension to a given service. The “uses” arrows determine services which use the output of another service. ...................... 22
3. This shows the different states the DreamZDash can enter into. As shown, the DreamZDash represents retains control of exit and entry into the system. ................................................................. 23
4. The diagram above shows the different integral processes required for the dashboard system. The orange outer-components represent processes. The blue inner-components represent classes or objects. These build up the necessary data-flow for the dashboard system. ................................................................. 24
5. The sequence diagram shows the order of calls and an example of interchange. In this example, the program starts up from scratch. The user then tries to change a few options before launching a game. Some of the options are respected, but the transmission occurs sequentially. ...................... 25
6. This UML class diagram depicts the necessary data-types for the game’s mechanics. The character provides an ample class used by all “living” entities. This means characters that have a “living” state and a “death” state. The player has these and a set of abilities denoted by it’s ability list field. ................. 28
7. The Viewpoint Component Diagram displays how each component produces into another viewpoint. This ends with the base-plate DreamZBox 2.0 main operating system or Linux. All viewpoints presented will build upon this system. Each viewpoint possibly provides to one or more other viewpoints. Viewpoints pointing to the same viewpoint may also interact with each other. .......................... 29
The UML Case Diagram for all actions a player can take in Maze of Dreamz during the game play. The player should be able to have interactions with monsters, items and check points as it is listed in the diagram.
FOREWORD: RELEASE NOTES 1.0

While the team is proud of what we have accomplished developing the Maze of Dreamz game, it is far from finished. A number of key features remain to be implemented, including:

- paths between rooms
- a wider selection of abilities for the player character
- more variety in monsters to fight
- objectives to progress through the game

Maze of Dreamz currently is an expandable graphics engine that runs in a web browser. Initially, the Qt framework was researched and found to be a viable means of writing graphics code for the DreamZBox 2.0. Unfortunately, this decision led to a major pitfall. Coding was being done on the QEMU emulator, not on the physical device, and while most of the console hardware was properly emulated, the graphics processor was not. Qt graphics libraries rely on OpenGL ES which is written to run on the GPU. Any future team wishing to migrate from web-based graphics should be aware of this technical limitation.

1 INTRODUCTION TO PROJECT

The DreamZBox 2.0 project is a Computer Science (CS) and Electrical and Computer Engineering (ECE) combined discipline project. DreamZBox 2.0 is a custom video game console, with its own controllers and video game(s) created specifically for the console. Requested by two Oregon State University Graduate students, the DreamZBox 2.0 represents a step up from the original DreamZBox predecessor. Additionally, the console itself will have its own custom controller. The gaming console is portable, rechargeable, and fun to use. Using Maze of Dreamz, CS29 Dreamz Catcherz, the CS team, show users the capabilities of the console and its interfaces. The system also uses a dashboard called the DreamZDash. This dashboard acts as a hub for the users. These software solutions aim to create an enjoyable user experience for the DreamZBox 2.0 console, allowing a fun video-game experience.

1.1 Combined Discipline

The CS team devised the virtual side of the project. This includes the user interface for the system, and the exclusive video game or games. The ECE group will handle the custom controllers and the console format. Both the CS and ECE groups will meet in the middle when working on the firmware for the console. This means that one of the most important things for this project will be maintaining an open connection between the ECE group and CS group.

1.2 Client

Corinna (Cory) Brown and Megan McCormick are two graduate students at Oregon State University. They acted as clients for the DreamZBox 2.0 project. They offered guidance and exposure to the video game development world. The whole group is pretty nerdy, with a deep passion for video games. Both clients worked on the original DreamZBox. Their project did quite well but failed in certain areas of development. They passed off a clean slate to a new group with hopes to guide individuals passed where they stumbled. The clients aimed for a market-ready solution while pushing team members to work for something they were proud of.
1.3 Members and Roles

The Computer Science team, aka CS29 CS29 Dreamz Catcherz, consist of:

- **Maxwell Evdemon**: ECE Liaison and Bluetooth Mastermind
- **Cory Hayes**: Discord Channel Master and Graphics Guru
- **Haofeng Tian**: Branding Manager and Game Developer
- **Tristan Hilbert**: Point of Contact and Dashboard Architect

1.4 Current Events and Outcomes

In March 2020, states and schools began mandating operations for students to stay indoors. This led to obstacles for projects like *DreamZBox 2.0* where physicality to the console was important. This also meant that an in-person demo was impossible for *EECS Expo*. So, only the game became available for demo due to the software constraints on the Dashboard and Bluetooth systems. This also meant that development slowed. Certain aspects of the game were never developed due to the slow development of the project, even with extended due dates for the project. The group did well to power through the ramifications and finish off the project however they could. The “outbreak” did inevitably harm the progress of the project.

1.5 Contributing to the Project

Future contributions are made possible based on the project scope. If future individuals wish to contribute or build off of the project, the repositories are available with client approval. Appendix 1 lists links for the repositories alongside descriptions of how to install the perquisite software. Sections 12 and 13 also provide beneficial resources to contributors. Additionally, documentation placed in Appendix 2 further derives the artistic direction of the project. The current state of the project still allows for a variety of continuations.
2 Requirements Document

2.1 Introduction

The DreamZBox 2.0 is a hardware system with portable components. It is a home use video game console. Additionally, it remains battery powered, allowing individuals to transport the hardware while in use. The users are given controllers with a joy-stick, buttons, and accelerometer to interact with the consolidated hardware. A software system is desired to provide the entertainment aspects for the DreamZBox 2.0. This software system must adhere to certain goals and functionalities listed in this document. The overview, requirements, and metrics are provided herein.

2.1.1 Purpose

The DreamZBox 2.0 uses cost-efficient hardware to provide an achievable entertainment experience for a largely diverse audience. The software must allow the DreamZBox 2.0 to remain accessible to a wide audience. Consumers will want to retain a simple and entertaining experience when working with the hardware. Stakeholders seek use of a video game on the system. Providing a portable video game console with proprietary controllers, the console needs an innovating game to demonstrate the hardware. The software seeks to support DreamZBox 2.0 in marketing, usability, and distribution.

2.1.2 Scope

The software system will provide a user dashboard internally called DreamZDash. This will allow usability of the console hardware itself for technicality. It will also provide a platform for a set of video-game applications. Additionally, the software will be packaged with a game built into the user’s library. This game will be called Maze of Dreamz. The software will seek to provide usability for the largest possible extents of the system. It will provide use of all peripheral inputs to the system. The software will not provide systems level accessibility based on security and time constraints. The software will not provide usability for any audience beyond the non-technical users. The software will provide a user-tested entertaining experience.

2.1.3 System Overview

2.1.3.1 System Context

DreamZDash will represent the first experience users will see and hear upon powering on the system. It will abstract the system. It will provide launching of applications like Maze of Dreamz. Alternatively, Maze of Dreamz provides graphical animation based on user input. The game will reflect a top-down maze adventure game. The user will play as a singular entity and attempt to reach a conceivable end while avoiding obstacles within the game. The singular entity will have virtual actions that will help progress it forward, including movement. Each map for the top-down maze game will be procedurally generated, in order to try and give a unique experience every time the game is played. After completion of the maze, the player will achieve victory.

2.1.3.2 System Functions

The dashboard should provide ample user interface to reflect possible user questions about system status. The dashboard should remain fully accessible through the controller. The joy-stick, being a push-in joy-stick, should provide full traversal and selection. The dashboard will look aesthetically pleasing and denote system state, including battery charge. When Maze of Dreamz is launched, the joy-stick will facilitate movement while the buttons facilitate different assigned
abilities. The abilities and movement will allow surpassing of the objectives within the maze. Each ability will light up the lights on the controller a different color. The user will have to use the abilities to achieve victory. Additionally, *Maze of Dreamz* is a multiplayer game that will work with two controllers, one for each player. In the case of multiplayer the second player will function similarly to the first player, as they have full movement and abilities. The two players will share the screen view between their entities. Finally, the user may pause the game at any time with a possible menu button.

2.1.3.3 User Characteristics

The only user supported by this software is end-users or players of the device. The possible users per console is two. These users will have to use the system without immediate oversight or help. The portable nature of the system does not restrict it to a specific location, but most likely it will reside in an entertainment system or livable space.

2.1.4 Definitions

2.1.4.1 top-down

A game style that is meant to feel as if the player is looking down upon the playable map and controllable player characters. When viewing a top down map is typically a two dimensional area that where all the game-play takes place.

2.1.4.2 procedurally generated

Components within a application that generate when the process is initiated. These components use forms of patterns and statistics to provide randomized experiences.

2.1.4.3 joy-stick

A joy-stick or joy stick is an analog form of input which supplies data across a two dimensional plane. Depending on the direction this component is pushed, it provides a vector in that given direction.

2.1.4.4 throbber

A graphical swirl or animation that denotes operations occurring in the background. Necessary for large amounts of data pipe-lining.

2.1.4.5 frame

A singular render or expression of graphical change. A still image.

2.1.4.6 LED

Light Emitting Diode: these devices are tiny lights that work with current.
2.1.4.7 USB

Universal Serial Bus: In this document the term is used in place of a memory stick or “USB stick”. These are portable devices that can plug into accessible ports allowing data transfer and storage.

2.2 System Requirements

The following sections list the necessary requirements and necessary attributes for the software of the DreamZBox 2.0.

2.2.1 Functional Requirements

2.2.1.1 For DreamZDash

- Shall interface through the Bluetooth controller.
- Shall launch a game
- Shall display images and graphics
- Shall provide a software power switch

2.2.1.2 For Maze of Dreamz

- Shall use 5 buttons, a push-in joy-stick, a trigger button, a scroll wheel, LED lights, and an accelerometer
- Shall display images and graphics
- Shall modify displayed images based on user input
- Shall light up LED lights based on game state
- Shall provide a different action or ability for each button and joy-stick
- Shall provide menu transversal

2.2.2 Usability Requirements

2.2.2.1 For all of the DreamZBox 2.0

- A player should be able to use the DreamZBox 2.0 controller without discomfort.

2.2.2.2 For DreamZDash

- Shall look aesthetically pleasing
- Shall use readable text
- Should remain simple and intuitive
- Should have use without explanation
- Should answer all potential user questions
- May include a throbber for loading content

2.2.2.3 For Maze of Dreamz

- Will not require the player to contort their hand in an unnatural way
- Should engage the player
- Should encourage the player to use all controls
2.2.3 Performance Requirements

2.2.3.1 For DreamZDash

- Shall pre-load images for games
- Shall display at correct resolution for potential devices
- May protect against setting changes harming battery life

2.2.3.2 For Maze of Dreamz

- Shall operate at more than a frame per second
- Will operate at the same rate with multiplayer

2.2.4 System Interface

- DreamZBox 2.0 shall connect to devices through Bluetooth wireless
- DreamZDash and Maze of Dreamz alongside other games will have an initial function to start their processes.

2.2.5 System Modes

- DreamZBox 2.0 shall have 3 modes
- DreamZBox 2.0 will have an unreachable terminal mode
- DreamZBox 2.0 shall have a startup mode consisting of DreamZDash
- DreamZBox 2.0 shall have a Game Mode consisting of games like Maze of Dreamz
- Upon failure the Game Mode shall fall back to DreamZDash

2.2.6 Information Management

2.2.6.1 For DreamZDash

- Shall store configurations and chosen settings in non-volatile memory
- Shall store user downloads in non-volatile memory
- Shall keep all memory stored on this system
- Should not allow other devices to access this data

2.2.6.2 For Maze of Dreamz

- Shall generate data onto volatile memory
- May store player progress onto non-volatile memory which may include:
  - Entity State
  - Level Generation
  - Player Collected Items
- Should not allow other applications to access this data
2.2.7 Packaging

- *Maze of Dreamz* and *DreamZDash* shall be compiled and compressed into a singular file
- The file above may be loaded onto the console through the internet
- The file above may be loaded onto the console serially
- Other games may be compressed and provided over the internet

2.3 Verification and Metrics

2.3.1 Functional Requirements

ALL 5 buttons, the joy-stick, and the accelerometer on the controller and is used during the game play. The *DreamZBox 2.0* can use HDMI port to connect with other devices.

2.3.1.1 Use Cases

- Use Case 1: The user can launch *Maze of Dreamz* with the joy-stick of the controller.
- Use Case 2: The user can use each of the peripherals on the controller, causing something different in *Maze of Dreamz* or on the LED lights of the controller.

2.3.2 Usability Requirements

The user should feel satisfied with game settings.

2.3.2.1 Use Cases

- Use Case 1: A new user want to experience the *DreamZBox 2.0*. They can understand the control of the console, select and play the game without others’ help.
- Use Case 2: The user start playing a game on *DreamZBox 2.0*. While playing the game, the user will feel satisfied with our default settings for buttons.

2.3.3 Performance Requirements

The *Maze of Dreamz* can continuously run on the *DreamZBox 2.0* for at least 6 hours without any battery charge.

2.3.3.1 Use Cases

- Use Case 1: The user can play *Maze of Dreamz* after a recent charge unplugged for 6 hours.
- Use Case 2: The user start playing a game on *DreamZBox 2.0*. While playing the game, the user will feel satisfied with our default settings for buttons.

2.3.4 System Interface

An interface will be provided to user to select games, change system settings and shut down the machine.
2.3.4.1 Use Cases

- Use Case 1: The user boots up the DreamZBox 2.0. The main interface will be shown to the user when the machine is fully boot up, then the user is allowed to select games, change system settings or shut down the machine in the main interface.

2.3.5 System Modes

Results of a tested environment using scripts will be provided to the client, detailing the restricted nature of the two modes. This will log the mode of operation before and after many different likely operations within code. Success will be shown if the modes only transition between modes is upon “launching into a game”. The system should revert back to DreamZDash upon failures. The tests will also screen for erroneous behavior such as DreamZDash failing. The features will deem complete upon success of these tests.

2.3.6 Information Management

The DreamZBox 2.0 will read the data from controllers and generate string type data files for user to store their system settings and game saves if necessary.

2.3.6.1 Use Cases

- Use Case 1: The user is playing Maze of Dreamz. They can choose to “save and quit”, and a save file will be generated by Maze of Dreamz to record the state of the game

2.3.7 Packaging

The stakeholders will have access to the compressed digital file. Any computer may download the software on their own. The software may also be emulated, but such verification will not be supported. Instead the publicised existence of the file will represent the accomplishment of packaging the software.

2.4 Closing Statement

The DreamZBox 2.0 alongside the DreamZDash and Maze of Dreamz will provide an entertaining experience to a diverse audience. The developers here at CS29 Dreamz Catcherz are really excited to get started. They share the vision of innovation the DreamZBox 2.0 can carry ahead of other iterations. The innovating factor within Maze of Dreamz is its creation of adventure puzzles rather than simple geometric maps. CS29 Dreamz Catcherz feels this game will pack a punch for the DreamZBox 2.0. All in all, the CS29 Dreamz Catcherz are gracious for the opportunity to conquer this project.

2.5 Changes in response to COVID-19

2.5.1 New barriers to completing the project

- ECE counterpart will not have final version of hardware for our team to integrate with
- Only platform independent part of project can be demoed (game, but no dashboard/bluetooth controller) without hardware
- Only one teammate with access to original Raspberry Pi prototype console
2.5.2 Plans to overcome these barriers

- Would like the code freeze date extended by 2 weeks to plan for virtual expo
- ECE counterpart no longer required to complete hardware, so testing framework needs to be adjusted
- No usage statistics can be gathered
- Requesting leniency in grading due to new requirements

2.5.3 Potential issues for delivering outcomes

- Attendance at virtual expo
- Bugs in “as is” prototype hardware from ECE team

2.6 Changes in response to Client Meetings

- The System Life is no longer necessary.
- The Packaging scheme is no longer available with due to outbreak. Hardware cannot be accessed.
- Packaging will now occur in the application posted online.

2.7 Appendix

Assumptions and Dependencies

The group assumes the following for the project:

- The DreamZBox 2.0 is a portable piece of hardware which can compute images on to a screen
- The DreamZBox 2.0 will be packaged with at least 2 controllers with a push-in joy-stick, 5 buttons, and an accelerometer.
- The team will have access to the hardware before release and the end of project.
Fig. 1. The DreamZBox 2.0 Gantt chart displays the plan for the next several months in implementing the viewpoints. The Gantt chart also includes other tasks which may take precedence during development. The Gantt Chart establishes the latest possible deadline for the given pieces of the project.
3 DESIGN DOCUMENT

3.1 Introduction

The DreamZBox 2.0 is a video game console, made cheap and portable. The console will host the game, Maze of Dreamz, created by the CS29 Dreamz Catcherz. Maze of Dreamz represents a playable experience to express the capabilities of the DreamZBox 2.0 console. The DreamZBox 2.0 console will also host a dashboard called DreamZDash. These software solutions aim to create an enjoyable user experience for the DreamZBox 2.0. The following sections work to uniquely identify this document. The background, contexts, and concerns are provided. The formalities and terminologies are also defined in these sections. These details make the project and software more understandable to a larger audience. They aim to provide better understanding before detailing the designs in the body of the document. All design concerns must be established to fully portray the competency of the software system. These can be magnified by examining the scope provided in the next section.

3.2 Scope

This document will focus on design aspects of an atomic software system. Breaking down the solution requirements, the concerns are grouped and defined in the body of the document. Software design views aim to meet sets of concerns based on chosen design viewpoints[1]. The views explain which concerns the viewpoint meets. The viewpoint dictates which concerns the view can meet. It also demonstrates the necessary elements needed to implement its view. The elements resemble the required tools, technologies, and structures. All concerns are met by one or more design views within this document.

3.3 Context

The designs work to encapsulate the software for a specific video game console. The DreamZBox 2.0 uses a proprietary controller with a customized interface. The innovative controller uses wireless hardware to transmit user input. Games on the DreamZBox 2.0 will make use of all features on the custom controller. The DreamZBox 2.0 will also come with a virtual hub (or dashboard). The dashboard will provide selections of settings and games for the user. Both the game and the dashboard adhere to the designs within this document.

3.4 Design Languages

The document produces the following design languages to detail the different viewpoints of our document. These all provide different conceptualizations of system components.

3.4.1 UML Case Diagram

A UML Case diagram exposes the external inputs and outputs to a viewpoint. It provides potential external actors or systems outside the system boundary. The diagram details the required services which the system must provide for external actors. These services are also known as “use cases.” The system or subsystem must contain each of these services. Some services have relationships with each other. One use case could help, extend, or cause another use case for a different actor. This language helps define the requirements of a system outside of its technological constraints. This leads to potential depictions of functional break-downs and data couplings.
3.4.2 State Transition Diagram

A State Transition Diagram details different necessary modes for a system. It classifies each mode then describes the transitions between them. Each state embodies certain objectives or services that must be performed until a transition occurs. This could organize the necessary objectives into possible applications or subsystems within a designed system.

3.4.3 UML Class Diagram

The UML Class Diagram is different than the UML Case Diagram. The UML class diagram provides structured formats for compositions of data. In other words, it denotes what each entity in the system must contain. Numbers, identifiers, and arrays, represent different data components which can attach to a given class. The classes can also expose relationships to each other. Some classes may contain other classes. Some classes may resemble other classes. These relationships simplify translation to code. This remains extremely true in object-oriented programming languages like C++.

3.4.4 Sequence Diagram

A Sequence Diagram provides a row of different systems or entities which transmit information to each other in a sequence. These are perfect for showing the transmission of data given an event or use case. The diagram exposes the relationships and dependencies between entities or systems. It also provides some functional decomposition and interface definition between coupled entities. Sequence diagrams, working together within a UML Interaction diagram, provide more definition to the system. For simplicity, a single sequence diagram is used to frame the general use-case.

3.4.5 UML Component Diagram

The UML Component Diagram provides an outlook on a given system’s composition. It breaks down the system into pieces. These pieces could translate into processes, classes, or other programming modules. These show the modularity of the design. It also portrays dependency and common structures within the design. This way certain programs may be used elsewhere in the code. The component diagram does not provide strict data-types, thus allowing large monolithic concerns to simply break down into manageable systems. The UML Component Diagram acts as an excellent format to decompose a system into programmable modules[2].

3.5 Views

The following views are given for the document.

- Hardware to Software Interface
- Dream Modes
- Dashboard Service
- Game Experience
- Game Logistics

3.6 Summary

The design for the DreamZBox 2.0 will focus mainly on points of interest. These are the case diagrams for the game, Maze of Dreamz, the dashboard, DreamZDash, and the hardware to software interface. Each of those three parts make up our goal of creating a unique game and user interface. The game’s design focuses on the game itself and the actions the user
can take within that game. The dashboard design deals with the start up of the console and the actions taken within the user interface. The hardware to software context design deals with inputs from the user, then the effects it has on the console and the software.

3.7 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth</td>
<td>A wireless connection interface using radio transmissions in hardware.</td>
</tr>
<tr>
<td>Disk</td>
<td>Slow long-term storage that maintains after the hardware powers down.</td>
</tr>
<tr>
<td>Driver (Software)</td>
<td>Software which provides decoding and use of serialized hardware data.</td>
</tr>
<tr>
<td>Hardware</td>
<td>The physical and electrical components of the system.</td>
</tr>
<tr>
<td>Interface</td>
<td>A layer of communication or transmission between the user and system.</td>
</tr>
<tr>
<td>Non-Volatile Memory</td>
<td>Data storage on a machine that retains integrity after the machine powers down.</td>
</tr>
<tr>
<td>Run-Time</td>
<td>The period where an application or piece of code is being used.</td>
</tr>
<tr>
<td>Software</td>
<td>The programmed components of the system. These components work upon hardware (see hardware).</td>
</tr>
<tr>
<td>Throbber</td>
<td>A graphical swirl or animation that denotes operations occurring in the background.</td>
</tr>
</tbody>
</table>

3.8 Design Stakeholders

Corinna Brown and Megan McCormick act as the clients for the DreamZBox 2.0 project. They worked on the original version of the DreamZBox. They want a prototype version of the DreamZBox 2.0. Alongside the Electrical Engineering and Computer Science (EECS) department of Oregon State University, they provide the funding for the project. They desire a functioning game and dashboard system on the perspective prototype system. The following requirements and concerns describe the aspects they seek.

3.9 Concerns

3.9.1 Fun Experience

The stakeholders want a pleasant experience within the DreamZBox 2.0. This includes a video during the launch of the console. Also, the console will provide an innovative game to spark curiosity for the console. Maze of Dreamz and DreamZDash must not hamper or create obstacles for the user. In all cases, the software should provide entertainment for the end-user.

3.9.2 Functional Interface

The software on DreamZBox 2.0 must function adequately. In other words, the graphics load from memory instead of disk. Secondly, the console adequately decodes input. Thirdly, the DreamZDash should show hardware status (i.e. battery life). Finally, DreamZDash and Maze of Dreamz need to allow transition between each other. All of these features must be accessible to the end-user. These should be in the form of graphical user interfaces manipulated by the Bluetooth controllers.

3.9.3 Descriptive Interface

This concern coincides with functional interfaces, but the user interface needs to be descriptive. The graphics used within DreamZDash and Maze of Dreamz should always show what is happening in the system. The user should not
need to question the current state of the system. Throbbers for loading and settings for graphics should allow for users to feel in control. The interface should include components like these to provide specificity within its design.

3.9.4 Multiplayer Game
The *DreamZBox 2.0* will be packaged with two controllers. This means the game needs to have a multiplayer aspect. The design should not assume a single playable entity. The camera, data, and mechanics in *Maze of Dreamz* should hold for both one and two users.

3.10 Hardware to Software Interface
This view considers the concerns towards device interaction of the *DreamZBox 2.0* with the external world. This mainly pertains to the Functional Interface. For this view, the best viewpoint would be a context viewpoint. This will abstract the system’s services from the external forces, framing the concerns dependent on external actors. This will help define where the software drivers need to exist in the system.

3.10.1 Design Elements
Any user or player will be considered a player within the use case chart. The output for video and audio of the system will be considered the television. The external storage for the game will be an actor called *Game Store*. The battery will be considered a Battery actor. The system responds to the different operations which can occur by these actors. The use cases within the diagram define these responses. The software must administer these use cases.
3.10.2 Design Language – UML Case Diagram

Fig. 2. The UML Case Diagram establishing the Inputs and Outputs of the system. It’s important that the external actors of this diagram could be translated to components in the DreamZBox 2.0 hardware. Also, the different arrows establish relationships between the provided services. The “extends” arrows provide an extension to a given service. The “uses” arrows determine services which use the output of another service.

3.10.3 Design Mentality

The design establishes the different services which must be provided to the external actors. The actors could resemble a hardware context. A player represents the Bluetooth input. The television represents the display port. The Game Store could represent the WiFi antenna. And, the battery could represent the battery sensor. Per context designs, the components do not matter. These actors were placed outside the boundary to define the inputs and outputs of the entire system. This way an operating system like Linux could be used to interface to all of the external forces on the system. Then, the system could reside within Linux, to facilitate functionality for these use cases. The design also shows lack of coupling between actors’ services, which allows for better performance within the design architecture. This format of design, allowing all services to act independently, will improve performance.
3.11 Dream Modes

This view takes into consideration the Dashboard/User modes portion on the DreamZDash. This designs around the concerns for a Fun Experience and a Functional Interface. The DreamZDash needs to provide a video upon launch. The DreamZDash must also provide the ability to launch a game. This viewpoint provides interface between the software components of Maze of Dreamz and DreamZDash for the DreamZBox 2.0.

3.11.1 Design Elements

The Video state represents a played movie file. During this state, the user cannot input. The user must watch the video. The Dashboard also uses the self-titled Dashboard state. This inquires the settings to be changed from the user. The user can launch a game from the Dashboard state launching into an In Game state. These states comprise all possible states within the diagram.

3.11.2 Design Language – State Transition Diagram

Fig. 3. This shows the different states the DreamZDash can enter into. As shown, the DreamZDash represents retains control of exit and entry into the system.

3.11.3 Design Analysis

The DreamZDash overlays the start up of the DreamZBox 2.0 and the process the user goes through when selecting a game. It also dictates the transition into further states. This means the DreamZDash will have some control over the orchestration of the system. The design organizes the necessary outputs for each mode. It also dictates which domains each software portion covers. The Video state will not have control over the operating system settings like the Dashboard state. Overall, the design well defines the order and organization of the different software states.

3.12 Dashboard

This view depicts the necessary programming blocks to provide the DreamZDash. A dashboard within an operating system is extremely hard to describe. The user will need an understandable but abstracted view of the system. The user must experience graphics within specific contexts. The graphics could come from games, settings, or hardware in
the console. The viewpoint governing this view must organize how it obtains and stores this data for graphics in a meaningful way. This view provides to the areas of a Fun Experience, Descriptive Interface, and a Functional Interface. By acknowledging how the data must be collected and maneuvered, the Dashboard view requires a composition viewpoint.

### 3.12.1 Design Elements

To allow for better translation into code, the orange outer components represent processes. These host a number of object instances, which represent the blue blocks. This excludes the *DreamZKernel* which acts as a singleton-class-process. Please refer to the design analysis in section 3.12.4 for further context. The relationships are also denoted between classes. The diagram’s circular inputs meet to socket-looking outputs. The relationships show a general movement of data from one class to another. Overall, the UML Component Diagram provides an insight into a possible process breakdown during the run-time of the dashboard.

### 3.12.2 Design Language – UML Component Diagram

Fig. 4. The diagram above shows the different integral processes required for the dashboard system. The orange outer-components represent processes. The blue inner-components represent classes or objects. These build up the necessary data-flow for the dashboard system.
3.12.3 Design Language – Sequence Diagram

Fig. 5. The sequence diagram shows the order of calls and an example of interchange. In this example, the program starts up from scratch. The user then tries to change a few options before launching a game. Some of the options are respected, but the transmission occurs sequentially.

3.12.4 Design Analysis

The DreamZKernel represents the parent software of the entire DreamZBox 2.0. In other words, the DreamZKernel should always run on the system, orchestrating the running processes at all times. This means the process must require as little memory as possible. The DreamZKernel will also keep track of the Dream Modes mentioned above, but the DreamZDash-Display will be in charge of playing the video during asset load. The DreamZKernel will be in charge of launching other programs. The order of process launching should be the following:

1) DreamZDash-Display
2) DreamZDash-ConfigurationManager
3) GLM
4) DreamZDash-HardwareStatus
While the sequence diagram shows the fine order of these messages, the processes should not wait until a message is received. Each process should use their old data until replaced with new readings or returns. The sequence diagram only works to clarify the orchestration of potential functionality within the dashboard system. In summary, the diagrams provide the best general use for the envisioned dashboard. Some patterns within the composition diagram could use further definition. These patterns of components are defined below.

3.12.4.1 Loaders

Many of the files and data loaded during the console's start-up will be organized beforehand. This shows the benefit of using non-volatile memory. The "loaders" will have a designated filename or address to always be reading or streaming data from. This means each one will have a dependency on the Linux operating system not shown in the diagram. It is recommended that these classes have a caching form or programmed container within their functionality. This will increase performance and speed since much of the graphical concerns require these data components.

3.12.4.2 Inter-process Exchange Coupling

In particular, the Foreign Process Exchange component depends on data from all other processes. This data coupling is abstracted, leaving some simplicity in implementation. The implementation of the Foreign Process Exchange object should be handled with care. With research, the recommendation from the designer is to use shared memory within the POSIX standard[3]. The Boost libraries within C++ provide the same ability [4]. The component will require extensive error handling. The implementation should always consider situations where all other processes do not exist. The DreamZDash should have an expected function without its dependencies met.

3.12.4.3 View Orchestration

Even after design, the View Orchestration component will need immense decomposition. The dependencies on graphical interfaces should be denoted later in design. This component is abstract in this diagram, because the chosen libraries like QT should provide all necessary dependencies. It should also be mentioned, that the Assets Cache and View Orchestration will heavily couple. The View Orchestration cannot display without the assets. This information is important for concerns within implementation, verification, and profiling. Should the View Orchestration object be broken down, perhaps the grouping of functions or component objects could allow a fast path for showing graphical media like the video.

3.13 Game Experience

The game experience derives majorly from the game's core mechanics. The game mechanics provide the important architecture for a game's design. While this document has intense focus on software, the mechanics of the game provide description on player experience. These respond to the concerns of Fun Experience and Multiplayer Game. By defining how the game should play, the software can follow the algorithmic procedure.

3.13.1 Design Elements

The different use cases within the diagram represent different actions for the player. These are game mechanics or functional descriptions necessary for the game. The mechanics describe the experience for the player. As they play through the game, the software must communicate these experiences through graphics and sound.
3.13.2 Design Language – Use Case Diagram
See section 3.17.1 for the Use Case Diagram.

3.13.3 Design Analysis
The Maze of Dreamz overlays the whole process of a player starting up Maze of Dreamz, playing it, and ending the game. At first, the player has to power up the DreamZBox 2.0, then select the Maze of Dreamz in the DreamZDash to start the game. During the game play, the player can interact with monsters, items and checkpoints. A battle event will occur when the player gets close to a monster. He can use the attack button to deal a small amount of damage immediately or use the skill button to cast some special skills after a certain time. The monster will automatically attack the player when he gets close enough. Items can be picked up on the map and be consumed in the bag. When the player has no less than 1 key in the bag, he can use the key to pass the locked door. The checkpoints can be set by the save button. The player can teleport to the newest check point anytime he wants using the teleport button. When the player loses all hit points, he can also revive at the nearest checkpoint.

3.14 Game Logistics
Data architecture remains integral to the project. The game requires organization of the game’s various data fields. These will conform to objects. This view provides to the Functional Interface and the Multiplayer Game concerns. This viewpoint respects the requirements for encoding the game into software. The necessary containers for data need to be defined for the best implementation.

3.14.1 Design Elements
The design elements represent the class hierarchy the game programming will follow. C++ will be used, so the different classes translate straight into the language. The diagram also provides “extends” relationships which mean the origin of the arrow uses all of the fields at the destination. The “relation” arrow describes an class being used within another class. In other words, one of the fields in the destination of the arrow is defined by the origin of the arrow. These relationships and diagram reveal the necessary objects for the designed game.
Fig. 6. This UML class diagram depicts the necessary data-types for the game’s mechanics. The character provides an ample class used by all “living” entities. This means characters that have a “living” state and a “death” state. The player has these and a set of abilities denoted by its ability list field.

3.14.3 Design Analysis

The character seems to represent a “living” datatype which could be used in a majority of the program. This will classify really any object instance which can move. It is shared with the Pos or position field depicting a tuple for two dimensional position. This could also be considered a coordinate on a two dimensional plane. The player will then have functions we could attach to input interrupts with an events module. Overall, the classes listed in the diagram provide necessary implementation for the game.

3.15 Design Rationale

3.15.1 Viewpoint Interaction and Rationale

For the DreamZBox 2.0 the viewpoints interaction is as follows. The Hardware to Software deals with player interaction. That player interaction is then used to decide the state of the dashboard, which can be seen in the Dream Modes viewpoint. These transitions control how the user changes the state of the DreamZBox 2.0 and the DreamZDash. On a deeper level, the viewpoint for the states of the DreamZBox 2.0 interacts with the Dashboard component viewpoint. In which the viewpoint makes sure to load and display the correct data based on the current state of the DreamZBox 2.0. When the state of the DreamZBox 2.0 proceeds into the Maze of Dreamz, the component viewpoint directs to both the Game Experience viewpoint and the Game Logistics viewpoint. This component viewpoint for the Dashboard suggests how to load the correct stored data based off the state of the user in game. The Game Logistics then provide the programming translation to the game mechanics within the Game Experience viewpoint. The rationale behind these viewpoint interactions is to make sure the
possible options of the player and the code line up with one another. In this case, the Hardware to Software components deal with player interactions which can change the state of the DreamZBox 2.0, thus changing what needs to be loaded for the component viewpoint. Then, the loading will include the data for the Game Logistics which embody the Game Experience. In that sense all the viewpoints are connected, each representing a different section of the DreamZBox 2.0’s functionality and building upon one another.

3.15.2 Viewpoint Component Diagram

![Diagram of Viewpoint Component Diagram]

Fig. 7. The Viewpoint Component Diagram displays how each component produces into another viewpoint. This ends with the base-plate DreamZBox 2.0 main operating system or Linux. All viewpoints presented will build upon this system. Each viewpoint possibly provides to one or more other viewpoints. Viewpoints pointing to the same viewpoint may also interact with each other.

3.15.3 Design Reflection

The necessary concerns for DreamZDash and Maze of Dreamz requires orchestration of process and data. These design choices define how the code needs to accomplish the concerns. Future designs outside this document will provide more detail in how each function will process. The processing of data remained the largest question during this design. The designs needed to plan the organization, extraction, and use of all kinds data. The team also wanted to view this project from a variety of angles, allowing for multiple types of viewpoints to be taken. The designs answer questions lacked in research. In conclusion, these designs work to define the necessary components for the many required tasks on the DreamZBox 2.0 running the Linux operating system.

3.16 Conclusion

This document covers the different parts of the DreamZBox 2.0 that our group is in charge of. This will mainly include the software sections of the DreamZBox 2.0 along with some of the firmware connectivity. This document specifically deals with UML figures and dictated overviews of each part. This includes the User interface dashboard, the firmware and software connection context, and design for the Maze of Dreams. We have explained the requirements for each of these sections along with analysis of these parts. From this, we can start to draw conclusions on our own design ideas, all of which will be used when succeeding in our groups part of the DreamZBox 2.0 project.
3.17 Appendix

3.17.1 Maze of Dreamz Flow Chart

Fig. 8. The UML Case Diagram for all actions a player can take in Maze of Dreamz during the game play. The player should be able to have interactions with monsters, items and check points as it is listed in the diagram.

4 TECH REVIEW – TRISTAN HILBERT

4.1 Introduction

This paper details the researched and chosen projects of the DreamZBox 2.0 Project. DreamZBox 2.0 is a video game console, made cheaply and portable. The console will host at least one video game created by software. The software
will use multimedia processing and user input. The following tools are required for these technologies: sound data management, software design pattern, and software multi-threading management. Many different versions of these tools exist in forms of technical solutions. These solutions must be usable on the given console system. They must not have dependencies past what is already decided in the project. And, they must provide the best performance in the project. The following sections provide more detail on the solutions.

4.2 Sound Data Management

Music provides fun background noise for users. When playing a game, users enjoy listening to soundtracks. It helps immerse the player into the video game. To play music in a video game, sound files must be decoded and streamed. These tools work to provide a system which can decode recorded sound files. This would allow for the team to include musical soundtracks and sound effects in the game.

4.2.1 Criteria

The assumed programming language is C++. Technically, the language the game will use is uncertain. C and C++ libraries are chosen because they can work with most languages as a shared library. A shared library compiles and runs outside the boundaries of the project, providing interfaces to the project. This allows developers to program in Java (for example), but still use the libraries.

4.2.2 Choices

4.2.2.1 Simple DirectMedia Layer: Sound Library

The Simple DirectMedia Layer libraries present ample cross-platform code for a C/C++ multimedia engine. Additionally their sound library provides use of many encoding formats[5]. These formats include MP3 and WAV. The SDL sound library does require licensing which requires legal efforts in distribution. The software is under GPL license which means distributors must release the code to clientele. The clientele are free to use the code in the same fashion. The software is also completely dependent on the Simple DirectMedia Layer library. It cannot be compiled without the library. This software has not been tested on the console’s hardware systems. This means executing the code could fail. Overall, the Simple DirectMedia Layer sound library is a solution with significant drawbacks.

4.2.2.2 OpenMP3

Alternatively, the OpenMP3 open source library decodes music files. OpenMP3 only works with MP3 encoded files[6]. This means music generation must encode the sounds into an “MP3” format. Additionally, this software is small. It allows for faster compilation of code (technically only beneficial to certain languages like C++). It’s interface makes it simple to put into a project. It remains completely atomic, meaning it does not depend on other code. The software contains an “unlicense” allowing use anywhere without restriction. OpenMP3 also lacks internal memory consumption. Sadly, this is only an MP3 decoder. It does not provide decoding for any other type of file. Overall the solution fits perfectly, but denies use of unlicensed sound encoding.

4.2.2.3 Creating our own decoder with C++

The blogpost by Bjorn Edstrom[7] perfectly describes how to decode MP3 files. Other tutorials could explain the decoding of files like WAV and OGG. These articles provide adequate research to create decoding software. This
would deny any need for licensing. Allowing personalization of the code, the project could interface more cleanly. The developers would have more knowledge of how the software worked, being the developers. The problem remains in the sheer amount of time this would take. Using a software library would allow for better use of time.

4.2.3 Discussion

The sound library for SDL (Simple DirectMedia Layer) provides the most versatile interface. It does not compare to the simplicity of OpenMP3. OpenMP3 has a license with more freedom than SDL’s sound library. Since building the library would take a long time, the OpenMP3 library seems most ideal. While encoding in only MP3 would remain a restriction, music generation typically allows the encoding in most personal computers[8].

4.2.4 Conclusion

The most optimal music decoder for sound would be OpenMP3. This software could compile into a shared library. This would provide background use for the software. It would be included on the console device. Most likely, the package for the game will link with the compiled OpenMP3 library. This way all the software running on OpenMP3 can make use of the simplified interface. A great link to sound drivers within the console[9].

4.3 Software Design Pattern

Adequate software design remains extremely important. To stay concise, the domains of design have widened greatly. In most cases, it would be better to abstract hierarchies of design, but this solution aims to create a standard of design for the project. This includes an architectural pattern. An architectural pattern provides the base relationships of the system. At the highest point in the software (and at the very core) the software works in an architectural pattern. While abstract, the core design decides many of the obstacles later on in the project. Design decides performance and opportunity. It also contributes to readability. This tool remains one of the most important in the project.

4.3.1 Criteria

Firstly, the design needs to provide the best efficiency of resources. A good design leads to speedy software. This will increase entertainment of the software. Secondly, the design also must refrain from too much complexity. If the design allocates too much work, the developers will fail. Thirdly, The design must respect object oriented and imperative based languages. Functional programming remains beyond the team’s ability. The design must make use of an imperative state, allowing for simple translation to code. In summary, the design must perform a simple yet effective use of a language like C++.

4.3.2 Choices

4.3.2.1 Layered Design with Object Oriented Programming

This remains the de facto standard for software engineering within introductory courses at Oregon State University. Separate all pieces of the software into atomic objects. These objects relate to each other via composition or inheritance. These objects make up modules. The modules can layer upon themselves to create different systems[10]. The layers of software provide a tree like structure in abstract. The programs remain readable and describable. The code is really organized. This leads to the biggest benefit of Object Oriented Programming: simplicity. However, tree structures in programming are slow. While the code remains simple to read and understand, it does not apply cache coherency or
state maintainability. These aspects allow the processing components in a console to work faster. Without these aspects, the code runs slow.

4.3.2.2 Event Driven Programming

Event Driven or Event Based programming[11] provides possible use of thread safe code. It allows for jobs or routines to stay separate. It allows for better transfer to net based programming. Despite the appeals to very efficient hardware capabilities, Event Driven Programming arises extreme coding issues. Side-effects, manipulations of the data, become extremely difficult to track down. This means code can provoke errors in other code. Data races, resource locking, and other technicalities would need consideration with this style of programming.

4.3.2.3 Entity-Component-System

This form of design carries a lot of nuance. Games like Overwatch[12] use this form of design for their software systems. Entity-Component-System or ECS[13] separates functionality from data. It works in three pieces. The (1)entities represent structures or objects which hold the data. The data represents vectors, strings, or flags attached to the object. These data fields represent (2)components. The underlying (3)system keeps track of which entities have which components producing correct manipulation of the data. The systems represent a sequence of jobs or routines that operate upon the entities’ components. There are much more adequate explanations, but this style of architecture allows for simplification of state. It dampens problems surrounding side effects. It also coordinates efforts for cache coherency and thread safety. Overall, the only draw back remains its nuance.

4.3.3 Discussion

Layered Design and Object Oriented Programming provide the simplest form of programming. Each developer on the team could understand it. It does not provide a benefit to performance like Event or ECS do. Considering performance of the software, in terms of speed and memory, ECS provides the best solution. It allows for extreme benefit in its organization of data and code. This explains the present use it has in games today[12].

4.3.4 Conclusion

The Entity-Component-System is the most prominent solution for design. It will require some introductory sessions for the team, but with the limited hardware capabilities, the team will need optimum power. Entity-Component-System provides the best uniform way of coding to allow measured forms of output. It also allows for multi-threading by its nature, mentioned in the next section.

4.4 Software Multi-threading Management

Multi-threading allows for ample usage of the entire processor on a machine. This means separating out work so each section of the processor does not interfere with each other. To abstract from hardware, many different software packages provide thread opportunities. Ideally the best package would be the fastest or the one with more functionality. However, the information below will only conclude versatility to differ between the three solutions and other tools might be necessary.
4.4.1 Choices

4.4.1.1 Simple DirectMedia Layer Threads

Since Simple DirectMedia Layer requires multi-threading for fast multimedia transmission, the library provides interfaces for its internal thread system[14]. This software does depend on the usage of the Simple DirectMedia Layer library. The usage of the interface only remains required when using the interfaces of the package. There are associated rules with using threads alongside the library. Since threads have access to certain pieces of memory, denying access to other threads, correct usage of these threads remains important. Sincerely, the package lacks atomic nature and provides less interface compared to other packages.

4.4.1.2 C++11 Standard Threads (GNU)

Thread objects have been made standard in all C++ compilers since version C++11[15]. This means the GNU compiler will provide definition to use of the objects. This would also be provided in MSVC and LLVM compilers. Should the project be programmed in C++, the thread objects provide an ample and cross-platform piece of software. This does have an atomic nature and it does provide the most versatility in comparison to other library packages. The standard threads also remain integral to all compilers which means no necessity for setting up executable environments or dynamic libraries.

4.4.1.3 POSIX Threads with Linux

POSIX[16] represents an IEEE standard for all operating systems. Linux distributions like Raspbian, the operating system for the console, provide full support for this standard. The POSIX standard provides its own form of multi-threading interface. These encapsulate as C functions called from libraries already on the system. These are dependent on the operating system, but since the target system has the dependency automatically, this is not a downside. The POSIX multi-threading subsystem provides the most bare-bones system for multi-threading.

4.4.2 Discussion

Honestly, these software have different names, but differ very little. In fact, each threading software probably depends on the POSIX standard. Since all solutions carry the same benefit, the best solution should be the simplest to use. Assuming the team selects C++11, the C++11 threads should be used. If another language is used, then the team should use the language’s multi-threading system. In general, research showed very little factual backing to performance differences between solutions. There were no developer observations or speed measurements to support better performance for any of these systems.

4.4.3 Conclusion

The C++11 Standard Threads represent the best solution. They require very little set-up and they have no dependency aside from language. They also make use of the POSIX thread subsystem among other optimized pieces[17]. The team should also use profiling. The research[17] explained profiling to have more effect on performance than multi-threading. Profiling allows developers to time functions in their code, recording the slower functions. Profiling[18] will allow better organization of subroutines and testing. Since the projects represents a real-time problem in software, a profiling tool would help immensely. While the threading system does not matter in terms of performance, the team should make use of a profiling tool for optimum performance.
4.5 Closing Remarks

As the previous sections mention, the team should use OpenMP3, Entity-Component-System, and C++11 standard threads. This assumes the team still desires atomic modules and highly free in their software. Should Simple DirectMedia Layer be used, the group may decide to opt for easier project set-up. One can only conjecture on the best forms of software for a given project. Some groups want easier facilitation and compilation times while others desire overall run-time performance. The metrics posted in this document expand beyond the necessities in prior requirement documents and problem statements. The metrics only seek to rectify a valuable product from the onset of the project. Concepts like the Entity-Component-System can implement into any project and still provide benefit. Hopefully, the importance of solutions like the Entity-Component-System will be held in higher priority than other solutions.

5 TECH REVIEW – CORY HAYES

5.1 Introduction

The core of our project is made up of many individual pieces. Our team has decided on a few key aspects to research. These include: art assets, user interface frameworks, programming languages, interfacing via serial port(USART), peripheral connection(I2C), HDMI, sound effects, design philosophy, multithreading, game engines, music generation, and image loading. I focused my research on where to find art assets, what frameworks are viable for building a graphical user interface for the console, and the programming languages to code in.

5.2 Art Assets

There are many websites offering 2D art for use in game development. The hardest task is finding art that we don’t have to pay for and is royalty free. The best sites I visited are: opengameart.org, craftpix.net, and itch.io.

5.2.1 opengameart.org

The selection of art at this site is varied and extensive. The site serves to not only showcase artwork, but to connect developers with the artists themselves through open forums. The diverse amount of content also makes the site difficult to navigate, however. My search queries would often yield the wrong results, and the tags used to filter them were inadequate. Also, it was often unclear what I was actually downloading. The file formats provided were decided by the individual posters, and there didn’t seem to be any restrictions.

5.2.2 craftpix.net

This site presented a much higher standard for quality. Many of the artists made their Photoshop files available for download as well, which could make customizing the art easier. Their search bar worked well and their interface for navigating the site was intuitive. Unfortunately, many of the assets were not free. The website also strongly encourages a membership with an accompanying fee.

5.2.3 itch.io

The design of this site is rudimentary but effective. Answers to my search queries were accurate and accompanied by relevant tags I could use to refine my search. It did take a little pawing through the results to separate the assets by quality. I had no sense of what the poster was offering until I clicked a link. In addition, each link led to a custom styled web page. There was no effort taken to stay consistent with the styles of the parent website. Consequently, some web pages were completely unreadable.
5.2.4 Review and Selection

While all of these websites offer a great selection of high quality art, I’m left unsatisfied if I can’t find exactly what I need. Opengameart.org tended to favor the artist rather than the developer. Most of the content displayed was discussion about the artwork rather than the artwork itself. In contrast, craftpix.net displayed its assets in a very accessible fashion. However, much of it was behind a pay wall. I’ve decided that itch.io will be our best candidate for 2D graphics assets. The time spent sifting through poorly styled web pages is recouped by a search engine that displays the proper results the first time.

5.3 User Interface Frameworks

The challenge is to find a multi-platform framework that is reliable, scales as our project grows, and is fast to implement.

5.3.1 LiveCode

This framework offers a very high level WYSIWYG (what you see is what you get) approach to programming graphical interfaces. It uses a proprietary language that would take time to learn, but coding in LiveCode is very efficient. Examples included replacing 11 lines of text formatting code written in Python with only 1 line of LiveCode. The framework also includes a syntax checker designed specifically for the language which is intended to prevent developers from writing “crash prone” code. The high level nature of LiveCode makes it less flexible to accommodate specific visual design choices. Also, many of the code editor’s features and extensions are pay only.

5.3.2 Qt

This open source framework boasts access to lower level features and is written in C++, which my teammates and I are comfortable programming in. At first glance, the learning curve for Qt seems steep. The documentation, however, is comprehensive and is written similarly to C++ docs. Being open source, the framework is also feature rich as the development community has taken responsibility for writing extensive libraries.

5.3.3 Kivy

Another high level framework, Kivy is open source and written using Python. It has seen wide scale use in mobile application development. Many of the application examples on the Kivy website are impressive. It is also OpenGL ready through extensions, which could be leveraged to develop visually stunning interfaces. Aside from showing some of the frameworks capabilities, the website was not very informative. I was unable to find any official documentation for coding using Kivy.

5.3.4 Review and Selection

Qt is the clear winner for our graphical user interface framework. First, it’s open source, so we’re able to develop with it, change it, distribute it, and have access to all of its features at no cost. Second, the development community is active in extending the frameworks features. Lastly, its documentation is easy to navigate.

5.4 Programming Languages

The choice of language to code in affects every other aspect of the game’s design. We’re tasked with choosing a language that integrates well with our implementations of graphics, sounds, and game controls. While the game may consist of code from multiple languages, my research is focused primarily on languages that have been used to code traditional applications on the Rasberry Pi.
5.4.1 C++
This object oriented language is probably the most familiar to everyone on the team, being that C/C++ are the first languages we learn at Oregon State. While it’s considered to be a high level language, in C++ we have access to pointers and thus polymorphism. It is also strictly compiled and not interpreted, so it won’t require an interpretive program on the console to run the game. Qt also relies on C++.

5.4.2 Python
This language makes it very easy to get a program up and running. Python tries very hard to limit the number of lines of code. This makes interpreting other’s code much easier. The language is also endorsed on raspberrypi.org. One failing for Python is that it can be slow when compared to compiled languages, which could be a concern when considering the limited hardware resources of our console.

5.4.3 Java
Java is a bit of a hybrid, relying on a just-in-time compiler (JIT) to run in an interpretive program called the Java Virtual Machine (JVM). The difference is that the JIT can compile down to bytecode allowing the interpreter to be much faster. Unfortunately, while the Raspberry Pi can run the JVM, none of the frameworks I’ve researched have been coded using Java.

5.4.4 Review and Selection
For familiarity and a robust selection of OO features, we’ll be coding our game mainly using C++. Our GUI will be done in C++, so for the sake of consistency and simplicity we should stick with it. It also doesn’t rely on an interpretive program, so C++ will not tax the console’s processing power as much.

5.5 Conclusion
We’ll have the easiest time coding a user interface using Qt for its robust list of features and the reliability and speed of C++. Following these same design decisions, we’ll be continuing to code in C++ for most of the other aspects of the game as well. Out of three strong picks for graphics assets, I’ve chosen itch.io for its ease of finding quality, free art.

6 TECH REVIEW – MAXWELL EVDEMON
6.1 Introduction
This document is part of the software side of the project, in which there will be four documents for each group member each researching a part of the project to make a general plan on how to accomplish our goal of making the user interface and unique game. The twelve research points split between the four groups members are 2D graphics assets, building GUI/framework, languages needed for different aspects of the project, the game engine, music generation, image loading, system sound, design philosophy, multi-threading, USART connections, I2C connections, and HDMI output functionality. My designated role on the software side is dealing with firmware, so the bulk of my part is making sure the signals sent form the controller and other hardware is properly recorded and displayed either in the user interface or the game itself. So this document will cover USART connections, I2C signal receiving and HDMI output strategies.
6.2 USART Connections

6.2.1 USART Introduction

USART stands for Universal Synchronous Asynchronous Receiver Transmitter, it is a communication interface that is used for transmission/communication between devices, in this case that would be our Bluetooth controllers. USART is preferred to UART as UART lacks the synchronous option, but can still use all of the other features of USART. USART is also what the ECE team decided on using for the communication between the controllers and the console. Since my job is primarily dealing with that of the software connections, this is something I need to decide on. The hardware we are working on is that of a raspberry pi, I will be making a choice between two primarily options by comparing them to choose one of those given libraries. these libraries are PL011 and MiniUART.\[19\]

6.2.2 USART Options

6.2.2.1 Mini UART

When working with Mini UART on the PI the baud rate for mini UART is generally tied to that of the core frequency of the VPU \[20\].Mini UART also typically is considered to be the “default when working with raspberry pi’s that have Bluetooth connectivity” \[20\] and will need to be reconfigured if the user wants to go with other available options. There are some issues with Mini UART that would be dangerous especially for a game systems like we are typing to make, in particular Mini UART can sometimes be to broad in the sense that it may miss some inputs from the user thus causing the player to have some dissatisfaction. However, Mini UART generally ”works good with linux” \[20\]. Mini UART also lacks some features that would be useful for data collection and handling, particularly when dealing with input signals. \[20\]

6.2.2.2 PL011

PL011 is typically the default when working with raspberry pi’s that lack Bluetooth connectivity. On the flip side PL011 is consider to be more reliable that that of Mini UART as it will drop less inputs form our controller, which in turn leads a better user experience with our console and the games on it. This lack of inputs drops is due to PL011 having a several FIFO inputs \[20\]. PL011 comes with several features that help it standout. PL011, has “break frame detection, framing error detection, the use of parity bits, receive and timeout interrupts, DCD, DSR, DTR, and RI signals” \[20\]. These are all pieces of information that can be valuable information to us as software designers, especially when detecting possible erroneous inputs form the user and what to do in situations where they occur. \[20\]

6.2.3 USART Option Comparison

The first thing that needs to be accounted for when trying comparing and trying to come up with a conclusion is what we are specifically looking for. In this case the most vital thing happens to input reliability, as we should be able to grab every single input made on the controller so no input by the user is lost. Unfortunately Mini UART happens to risk having some of the inputs drop on it’s connection while PL011 does not have this issue and is typically able to properly record every input made by the user. Mini UART lacks some capabilities that PL011. In this case those are the features I listed in the PL011 subsection, parity bits, DCD, DSR, ect \[20\]. the software CS group wants to work with a familiar operating system, in this case that operating system is Linux. This is something that Mini UART has over PL011 as it tends to mix better with Linux than PL011. \[20\]
6.2.4 USART Conclusion

In this situation we the users are only allowed to use one of the raspberry pi’s built in UART capabilities for receiving these inputs from the Bluetooth controllers. In this case due to that limitation the choice should be PL011 as it is better equip for error handling and in general, it also is more accurate with its inputs dropping less due to having more FIFO inputs [20]. The fact that Mini UART can drop inputs when we need them to be particularly precise is red flag as, and on top of that Mini UART offers no real benefits over PL011. The error handling aspect of PL011 is an important part as it allows us to prepare for cases Mini UART could not tell us. We will be using PL011 along with the WiringPi library to manage the serial inputs from the controller to software.[20]

6.3 I2C

6.3.1 I2C Introduction

I2C similar to that of USART in the idea that it is going to be primarily used for the communication, or the idea of “communicating any non serial inputs”[21] to the console which may need to be used for both the user interface and game play of our created unique game. “I2C is a two wired connection interface to used for low speed connections between devices like micro-controllers” [22]. Alternatively the option was the use SPI which deals with several more wires which is why the electrical engineering team decided to go with I2C as it primarily way of making connections. So working with firmware I need to make sure the software computer science team can work with those inputs properly. This decisions with primarily be dealing with c++ against python for which is better to be used to keep track of these connections. In this case c++ will primarily use the WiringPi library while the python version will be the group keeping track of it by themselves.[21][22]

6.3.2 I2C Options

6.3.2.1 WiringPi(c++)

WiringPi is often seen as the go to library for working with raspberry pi’s along side using I2C connections, this is due to being an already established library that was made specifically to work in this environment. the WiringPi library deals in C++, and while not difficult by any means, it still has more syntax than some other languages. This could result in spending more time dealing with syntax issues rather than the code itself. However, this may also increase the quality of the code as more time needs to be spent implementing that syntax.the library itself is written in C++ so it comes with the benefit of being able to use GDB as a debugger, which in turn will help reduced the amount of time debugging. Another benefit comes from the fact that the WiringPi library is already established is that there is specific functions within the library that will handle the general set up of the incoming information and process for the I2C inputs from micro-controllers [23]. This in turn could help save more time for the software team in the longer scope as we would have those pre-made functions.[23][21]

6.3.2.2 Configuring it Ourselves (python)

One of the bigger benefits in working with python is that it is easy to understand and use. It is also Rather easy to find bugs and errors within the code as we will be writing the functions ourselves to handle the code. This unfortunately means that it will probably take a considerable amount more time then working with the pre-established libraries, as we will be making it ourselves. This could result in generating better quality code as we will have to truly understand the
concepts this way, but it is still a big risk to take. Another known negative is that the python version has been known
to run considerably “slower than other” versions of the code [23], which could result in missing some inputs or at least
not catching them in time which in turn hurts our users experience. [23]

6.3.3 I2C Option Comparison

When comparing the I2C options it is important to first consider what is important and why, when making a choice.
In this case not recording or dropping inputs is one of the most major negatives. Along with that we preferably want
something that will not be that hard to debug in case of errors and not take a long time to create as well. The biggest
downside of using python to write all the functions ourselves is that it is noted to run slower than pre-established
libraries like WiringPi. While python is easier and fast to write than c++, as it is less syntax oriented, writing our own
functions will never not be a risk when we would already have access to them in another library. This would usually
make the python code easier to debug, but when taking into account debuggers like GDB this point becomes no longer
important. Another risk is that the library WiringPi may not have all the functions we need, but in that scenario we
would already be writing the functions from scratch anyways. On top of that since the functions for WiringPi are already
created it will also help reduce the time spent on this portion of the project.[23][21]

6.3.4 I2C Conclusion

Fortunately this a rather easy choice when taking things into perspective. As once again the most important thing being
run-time, that is if the code runs to slow we miss inputs and if we miss inputs that can hurt our user experience as it
is meant to be an entertainment console. For that reason alone it is enough to chose WiringPi (c++) as our primarily
library/code when working with these I2c connections. Another reason happens to be the amount of time we have to
work on the code. Writing code from scratch when a library exists to primarily deal with the job is not generally a good
time wise decision. This is because inputs are important and we need to have them done as soon as possible so we can
begin proper testing of the user interface and the unique game. This would also allow us to spend more time perfecting
and debugging that game as well, due to not needing to write our own functions.[23]

6.4 HDMI Functionality

6.4.1 HDMI Introduction

HDMI stands for “High-definition Multimedia interface and is primarily used to display both audio and visuals over
single line.” [24] This once again is something I feel as if fall under my job when working with firmware, as now
that we have created the software we want to then display that software so the user can see it. There would be no
point to to make any of this software without having any way for it to be seen. In this case this HDMI is the obvious
choice as it works with both audio and video data, along with being something that is directly integrated with some
raspberry pi’s themselves. When going to look at this would be done there is generally two different ways. For HDMI
functionality we can manually do everything ourselves using system code or we can use a built in library under the
name of Raylib.[24][25]

6.4.2 HDMI Options

6.4.2.1 Coding from Scratch

The downside of writing it ourselves is that requires a deep level of understanding and comprehension of the topics
at hand, which in turn could take quite a long time [25]. This is also another step that is rather important our software
process as this is where we get to test the graphic capabilities of our code to make sure they are functioning as intend, which means we may not have time to write out own code and deal with the frame buffers. Once again this process could result in higher quality code when compared to using a built in library, but such a chance is unlikely and time consuming. There also would be no risk of any created functions not working with each other, since all the functions would be written by us and we can edit them however we want. Which the same cannot be necessarily said about and pre-determined libraries.

6.4.2.2 Raylib

While Raylib does have that slight downside as mentioned, it also comes with all the benefits of a pre-established library meaning it does come with a large amount of premade functions that can be used to deal with managing the framebuffer. Another benefit that Raylib has is that is is “available on almost any platform” meaning that we should have no trouble working with it in any environment, or trying to render to any given screen that supports HDMI capabilities. Raylib also has no “outside dependencies” that we would have to worry about when worrying with it meaning no time would have to be spent trying to make sure all the dependencies are in place. Raylib is also “hardware accelerated”, this means Raylib should typically get good performance, which is key when the user has to look at the screen to primarily understand which is going on. This should in turn help with user satisfaction as no user would want to watch a screen that has some lag time behind the their input. The only real risk is that Raylib may not have functions we need to properly display the user interface and unique game.

6.4.3 HDMI Option Comparison

When comparing the options we need to first establish some the preferred criteria. In this case display is something that can be complicated at times we would like to not spend a large amount of time on as getting our device to properly display the software. This is due to how vital screen display is for both the entertainment console and testing the software on that console. As something could be in theory written correctly in code, but if it does not display right then that would still be considered to be a failure. However, the most important thing is that the code runs well, so there is no visual stuttering on the screen. With that said coding the HDMI framebuffers from scratch is something that could take a rather long time which could hinder our testing of the software. It can also be complicated to work with the framebuffers and without a deep understanding of how they work unnecessary mistakes could arise. The only real downside when working with Raylib is that it may not have all the functions we need for display, and that is still only a chance. Raylib is also “hardware accelerated” meaning there should be no real performance issues or stuttering on screen. That is something that cannot be guaranteed when working writing the code ourselves. Raylib should also help reduce the amount of time spent working on the assignment due to the pre-established functions.

6.4.4 HDMI Conclusion

With all the information we know that choice is rather easy. Not using Raylib for HDMI and framebuffer work would actively hinder us rather than help the group. Once again we need to keep on time to get the project done and taking a risk in trying to write our own code could turn out badly especially since there is already created code libraries that do exactly what we want to accomplish. In this case, Raylib is also “hardware accelerated” meaning that it would still most likely have a better performance than our written code and when working with something like a video game performance is always a key factor. Which is another reason to why Raylib is superior to writing our own code, not to
mention how integral visuals are for actually testing our software as well. All of that adds up to make using Raylib the best choice we could take for working with HDMI and graphical interfaces. [25]

6.5 Conclusion

After looking deeper in to topics about firmware and its associated parts I believe I have found the best possible choices for each of my three topics. I have come to the point that when working with USART the best choice for our group would be to use PL011, the best choice for working with I2C would be the WiringPi library, and the best choice for working HDMI outputs would be the Raylib library. I feel that these fall within my jurisdiction when working with the firmware for this project and are choices I wish to use when moving forward with my role.

7 Tech Review – Haofeng Tian

7.1 Introduction

The DreamZBox is a console with features that are unique to the console gaming market. This paper is going to do some research about technologies for certain aspects of the DreamZBox 2.0 project. For each technology, 3 methods will be provided to solve it, and at the conclusion, each technology will have a most suitable way to achieve.

7.2 Metrics

There are many different methods can be used to achieve a technology in our project, this paper will going do research for 3 possible methods for each technology, then compare through each method and pick out one best solution for each technology. If we can get a clear idea about what tools or methods we should use for the project, save time and increase the efficiency after the research, that would be the biggest success for this paper.

7.3 Technology

7.3.1 Music generation

7.3.1.1 Ableton Live

Ableton Live is a digital audio workstation for macOS and Windows. Live’s user interface is composed of two ’Views’ – the Arrangement View and the Session View. Live utilizes audio sample or MIDI sequences, referred to as Clips, which are arranged to be played live (i.e. triggered) or played back in a pre-arranged order. MIDI triggers notes on Live’s built in instruments, as well as third party VST instruments or external hardware. The Intro version of Live includes four instruments (Impulse, Simpler, Instrument Rack, and Drum Rack) and the Standard version of Live additionally includes External Instrument, with users having the option to purchase additional instruments.

7.3.1.2 Logitech Media Server

Logitech Media Server is a streaming audio server supported by Logitech (formerly Slim Devices), developed in particular to support their Squeezebox range of digital audio receivers. It supports audio formats including MP3, FLAC, WAV, Ogg, and AAC, as well as transcoding. It can stream to both software and hardware receivers, including the various Squeezebox models, as well as any media player capable of playing MP3 streams. Plugins from Logitech and third-party sources are also supported, allowing additional functionality to be added
7.3.1.3 Audacity

Audacity is a free and open-source digital audio editor and recording application software, available for Windows, macOS/OS X and Unix-like operating systems. It can import and export of WAV, AIFF, MP3 (via the LAME encoder, now integrated as part of Audacity), Ogg Vorbis, and all file formats supported by libsndfile library.

7.3.2 Game engine

7.3.2.1 LibGDX

Libgdx is a Java game development framework that provides a unified API that works across all supported platforms. The game editor can be run on Windows, Linux, Mac OS X, Android (2.2+), BlackBerry, iOS, Java Applet (requires JVM to be installed) and Javascript/WebGL platforms. For audio support, it have streaming music and sound effect playback for WAV, MP3 and OGG. It can use mouse and touch-screen, keyboard, accelerometer and compass to get the input form the user. And most of the graphic work is made by openGL.

7.3.2.2 Godot

This is also a free and open source for designing 2D or 3D games, the editor can be run on Windows, macOS, Linux, FreeBSD, OpenBSD and Haiku platforms. The language it use are GDScript (which is similar with python), C and C++. It also support some additional languages such as Python, Nim, D and other languages.

7.3.2.3 AppGameKit

AppGameKit Classic is a tool for making games for mobile devices. According to the introduction, it is designed fit to Raspberry Pi, it supports Android, IOS, Windows, Mac, Linux, HTML5, Raspberry Pi platforms. It mainly use C++ language. OpenGL can be used to create graphics.

7.3.3 Image loading

7.3.3.1 SDLimage

SDLimage is an image loading library that is used with the SDL library, and almost as portable. It allows a programmer to use multiple image formats without having to code all the loading and conversion algorithms themselves. It can load in the popular image formats such as BMP, PNM (PPM/PGM/PBM), XPM, LBM, PCX, GIF, JPEG, PNG, TGA, and TIFF formats. These are loaded onto SDL Surface and painted onto the screen as normal.

7.3.3.2 Piskel

This is a tool to create Pixel art, which is used in many classic games. It can use pixels to draw 2D graphics and short animations, by using different colors the image can also looks like a 3D image or animation.

7.3.3.3 Tiled

Tiled is a popular tilemap editor in open source game development. It is used to assemble consistent, retro-looking backgrounds for the in-game scenes. It can export maps as XML, JSON, or as flattened images. It is stable and cross-platform.
7.4 CONCLUSION

7.4.1 Game engine

All of three game engine are Multi platform, Godot mentioned that they have special support for 3D design, but we have decided to make a 2D game, so we don’t need that function. Only AppGameKit said that they have support to Raspberry Pi platform, I’ll going to pick AppgameKit as our first choice for the game engine.

7.4.2 Music generation

All of three music generation stations are free and open source, since Logitech Media Server is designed for Squeezebox series, that may not best fit to our project, after comparing Abletion Live and Audacity, I prefer the design of Albeton Live, so I will pick it for the music generation.

7.4.3 Image loading

We haven’t learned any skills for computer graphics at art level, Piskel may require us to spend a long time to analyze the use of it. I will choose SDLimage because so far we prefer import some free online materials for the graphic, the feature which can make fancy images will be added after the alpha version game.

8 WEEKLY BLOG POSTS – TRISTAN HILBERT

Blog Post

- October 18th, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

I have completed a fair amount of research and bonding within my first few weeks of progress. I have done research on the system for my team. I have gotten them to limit to a game design scope. The group will be doing a two-dimensional top-down mazer. I think personally, the project is a bit too limited. It remains easy to look forward to though. I am glad we have at least solidified in idea and objective. It becomes easier to solidify the rest of the project. Especially as the group seeks requirements and design choices, the project will need to come full circle soon. I think the group needs to settle for something sooner, rather than later.

Problems

I think the group will hit road blocks on design. I see the excuse of “we didn’t decide on this” come up to soon. The client will not always have all the answers. As engineers, workers must derive solutions and tests to further formulate the project. I think a Spiral workflow complements this idea. However, this is an agile workflow class. I also think it’s a weakness of the group. The client does not care what genre of game is created. The group was given a problem. The group only needs the problem. Continually working under the idea that the boss needs to design the work for the group will not work. I fear this will be a reoccurring obstacle.
Plans
I will have to reach out to several individuals over the next week. I need to buy time for my groups writing and creative process. At the same time, I will try and help out the ECE group with some of my own software. I have some free-time this weekend since plans got cancelled. Hopefully I can put some thoughtful work into my school work, amplifying the possibilities of doing things later in the term.

Blog Post
• October 25th, 2019
• Tristan Hilbert
• Project: DreamZBox 2.0
• Team: Dreamz Catcherz (CS29)

Progress
We actually got quite a bit decided and done this week. The group has committed to an adequately difficult task for the project. Everyone is on-board. Per deliverables, I contributed some pencil prototypes. I also corrected the requirements document. Cory and Haofeng finished the final problem statement. We are all starting to communicate as a team. I have been endorsing individuals to speak out in meetings. Every member has great ideas, but they don’t voice them over my mediation. I think they are improving.

Client Progress
We met with the client on Wednesday. We got our questions answered about platforms and requirements for the game genre and content. It seems the client genuinely seeks an open ended product. The only requirement they desired was innovation. Additionally they desired some functionality in the controller and in the dashboard. Mainly all peripherals including LEDs *must* be used. This is reflected in the requirements document. They really want the dashboard to look pretty. The ECE group mentioned a funny video they want to play on the onset. I left this out of the document, but we will most likely implement it. Overall, it’s free reign. Next week we are to present our game in an “elevator pitch.”

Problems
I am having problems working with the group’s apathy. It seems they see the project as a huge monolithic artifact. Yet, they want to program everything from scratch. I am fine, because I realize what is at stake. However, if we press forward with me doing most of the coding, the group will lack understanding later on. I just feel the group has not looked into this project in their free time. I think it would help for them to not treat this project as a set of assignments, but rather an opportunity for coding agency.

External Problems
It seems the ECE kids have much stricter requirements than us. This will cause them to need more personal time during client meetings. They have to ramp up their contributions much quicker than us. I think it would help if our group volunteered more efforts to help them rather than act apathetic. That or start on designing the game ahead of time.
Missed Meeting

This week for the TA meeting, only half the group showed. Both Haofeng and Cory did not understand the meeting was *weekly*. This was made PERFECTLY CLEAR in the emails. I want to blame myself for poor communication, but they goofed up. I was CRYSTAL CLEAR about what WE agreed to and why this was necessary. They are choosing not to pay attention.

Plans

I reached out to people this past week. In general I will be going slightly rogue. I have no doubt in my mind, that I will be doing most of the coding. I have yet to see any group member take initiative. At the same time we need to meet as a group to consider the ramifications of what we are planning to do. The individual assignment this week will help with that. I am a very data oriented programmer. I seek low complexity with high efficiency in code. I also seek high amounts of design to prevent the question “HOW DO I DO THIS” from other group members. So the following will be done in the next week.

- Meeting
- Research on possible technologies
- UML and ERDs
- Research on Game Engine Architectures
- Learn some basic ARM assembly
- Learn toolset for using gcc compiler

Blog Post

- November 1st, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress

I took the initiative to break the project into pieces on Monday. I presented the different pieces to my group so they could research into different areas. Since we are building this game from the operating system up, it will be important to know all the necessary pieces. I personally chose different aspects of optimization. These will keep our project current and fast. I partied a lot this week, so I did not get time to do everything I wanted to. However, My group did meet together on Thursday. I am happy most things worked out.

Client Progress

There was not an official meeting with the client this week. Our eecs counterpart did change the controller on us. It now has some new peripherals which will change the original pencil schematics a bit. The controller will now have the following.

- 1 Accellerometer
- 4 Main Buttons
- 1 Trigger Button
- 1 Start Button
- 1 Bluetooth On/Off Button (Not implemented By Us)
- 1 Speaker
- 1 Push-In Joystick
- 1 Scroll Wheel (Left Side Trigger)
- LED Lights in Enclosure

**Problems**

The controller redesign brings an issue. The overarching problem remains implementing each control. The group could keep adding features to the original design. This will hurt the simplicity of the project. It will add time required to the project. So, only adding features to equate the new inputs is not adequate. To respond to this dilemma, the group might consider a redesign to the game. This shouldn’t be too problematic, as the game is still an idea. The group needs to iron this out by the end of term.

**Plans**

Similar to last week I will continue research on my own. I am seeking out other systems to program and add on to the console. I will also need to come up with new pencil designs and maybe some UMLs. I want to have this game well reflect all the inputs on this controller. We should start small and build outwards. Following last week I will continue doing the following:

- New Pencil Sketch
- UML and ERDs
- Research on Game Engine Architectures
- Learn toolset for using gcc compiler
- Consider the porting of apk files or sb3 files.

**Blog Post**

- November 8th, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

**Progress**

This week we made a lot of decisions according to the game. Every input is now attached to an action in the game. I think this will further solve the problem. It will display the efforts of the electrical engineers adequately. We are also considering simplifying the project. Now that we have done tech reviews, we are considering adding in engines. Cory says he wants more time to complete his research. He is picking the language we are using. I also figured out solid architecture designs to use.
Client Progress

In the client meeting we confirmed the new controller. The previous progress report denotes the features. The client also made some previous firmware code available. This will help Max, for he is in charge of firmware.

I then asked if we could record in two meetings. We need footage for the final assignment of the term. It would help to film the meeting. The content could add footage for the project. The others suggested renting a camera from the library for the best quality.

To finish the meeting, we established “Quick Time Events” will be in the game. The meeting was shorter this week.

Problems

I have problems in my writing. My writing uses a lot of loaded terms. I need to work on being specific, talking through my writing. Kirsten recommended trying to transcribe rather than write. Also, I should recognize my audience. I should use terms that make sense outside of the project.

I really feel these techniques will help my writing. I have been receiving critique from my teammates and professors. I really want to improve my writing. With practice, I am sure, that I will get better.

Plans

I accomplished a lot last week. By solidifying the design, I accomplished many of the tasks listed last week. I plan to get the team ready to start coding. I think the other team members also want to create their own game. I hope to do the following.

- UML and ERDs
- Emulating the Raspbian OS on windows
- Making the emulation portable
- Consider the porting of apk files or sb3 files.

My team really wants the ability to test on their own machine. I am perfectly happy getting this working. It should only take a couple hours of my time. Once everything is set up, we can begin progress on the deliverables.

Blog Post

- November 15th, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress

This week I fleshed out hefty portions of the design. I had a blast learning through new frameworks and producing plausible design steps. I decided to officially lead the development of the Dashboard. This way I can control the bare minimum produced application. This will at least assure a platform for the ECE crew to work with.

From last week, I was able to find an emulator for the project. It limits memory usage, but it gets the job done. I hope it works for the team. It is a simple exe install. I would hope that does not complicate things too badly. All in all, everything is Linux based and quite resembles the engineering servers.
Problems

The design document was due this week. I really did not get a good set of designs for different aspects of the project. I did volunteer to do most of it, but I was expecting some input. In either case, I ended up completing most of the document myself.

I did mention; if the document were too overwhelming, then the group could change templates. I think this majorly distracted the group. I know Cory researched into a new template. I have yet to see any documents arise from it. I offered to email it to the TA even. I am really scared the team is no longer on the same page.

Plans

I plan to complete the design document as fully as possible. I also need time to complete my finals for other classes. That may take priority. I still think completion of the design document and the finals for other classes would provide a huge amount of success. I am deeply excited for what is to come.

Blog Post

- November 22nd, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress

This week I spent focusing on work outside of capstone. I have final projects with rapidly arriving due dates. I also had some opportunities to learn about Apple iOS mobile development. With these *distractions* there was little I did for the project. I did put some major energy into the Design Document 2nd draft. Max (Maxwell Evedemon) helped me out incredibly, taking on different segments of the draft for himself. This week I felt sympatico with my teammate. I also had a one-on-one meeting with the Teacher’s Assistant. This helped further my understanding of the requirements for the design. I finally understood this document to represent an broad encapsulation of the project rather than an ironclad blueprint. I kept a lot of the poor writing in the document, but I feel the new organization helps readability. I also included some new charts, further emphasizing the plans for the future. While there was little documentation, I really enjoyed the amount of effort put forth this week. There was a lot of communication, and that helps in unifying the team.

Client Progress

I sadly missed the meeting with the client, due to the one-on-one. The meeting was documented by Max. Max reports the following:

The ECE discussed their progress, struggles, and problems based on enclosures. They also gave updates on the hardware for the Raspberry Pi and Bluetooth controller. The CS groups reported investigation of splitting up the project, stating the intention to start the project sooner rather than later.

I then had Max ask a couple of questions for me based on research for the design document.

- Is the source code for the requested video on the Google Drive (I need to know the code’s composition)
• Is there a button on the controller to straight to the dashboard (Do we need to keep the dashboard running in the background of the game)

He then retrieved the answers for me. He established the source code is on the console, not the drive. There is a copy of the mp4 (video file) on the drive. He then reports the composition puts the video on the Kernel rather than GRUB. There is also an image which can be used for boot. Secondly, there will be a button on the console itself that goes back to the dashboard. Try to keep the dashboard up in the background unless start time is less than 10 seconds (It won’t be). It is ultimately up to us.

The information Max got is really really really helpful. It helps me realize further miniscule concerns missed in the initial design process. This means the launching of the game must not kill the dashboard process, but instead limit memory (most likely). This means the Dashboard will have a *suspend* state most likely. This way the game can use as much memory as needed.

**Problems**

I really wanted personally more time for design. I really think we could have gotten into huge amount of detail. This would have been safer for development. There is a big notion of “technical debt” or “feature creep” (https://www.youtube.com/watch?v=L5NfbpLmMHI). The game needs time to test and design. I really feel like the class is not emphasizing the need for design enough. This is based on the universal grading that is needed. Many other groups have projects without all the necessary details. Our group has a really large project. If we leave stuff to be decided, it could destroy the project. I hope in the coming months, the other team members will help me describe and define the game mechanics and logistics. This will prevent large amounts of design to occur at development.

**Plans**

The current plan is to finish finals for the next week and a half. I will push the progress report to Haofeng or Cory in the team. I will help, but I will not be overseeing the project this time. I will start on the dashboard in December, hopefully finishing the development ahead of schedule. This way I can talk to third parties about porting games to our system.

**Blog Post**

- November 29th, 2019
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

**Progress**

I finished my finals for Environmental Ethics. I should have my Computer Architecture final done by Monday. Within realms of the project, I have made minor communiqué. Due to the holidays, no one is expected to get work done this week. I don’t think it would be an appropriate requirement anyways. I would want everyone to enjoy their holiday. Plus this project will start development by finals week. We are on schedule. We are exactly where we need to be.
Problems

One thing that I am struggling with is micro-optimizations. When thinking through the code processes, I worry myself with the many dependencies the project will have due to time constraints. The Linux Operating System seems bloated for our efforts. We do need many of the POSIX operations catered by the operating system. I was talking about this with my Dad. He suggested to look into personalizing distributions. Overall, I think this would be a cool thing to look into, but having a severe dependency on a system makes it hard to replace. I will consider it for the project.

Plans

I plan to start the github repository this next week. This way development can start on the project. I hope to start on the project over winter break. I feel this will continue to bring the group on track. I also want to start some prototypes of the game in python or something quick. This way the group can have some design implementation and some user experience tests. I don’t know how much I can accomplish, but I am willing to look into it.

Blog Post

- January 7, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress

I did not make as much progress on the Dashboard that I wanted to last week. I spent a major amount of time on the project, amounting to about 20 hours. I figured out how to port the current features to the Raspberry Pi. I also abstracted some of the graphical features into C++ objects. However, I hit some problems on resizing the window. Eventually I participated in meetings halfway through the week.

So, I attended the client meeting where the groups gave progress reports. During this meeting, I had broken my graphics code. It turns out I was missing function calls to actually implement the graphics API the second time around. I did not have anything to show. I did have code and repositories to demonstrate my work. I lacked a serious demo. It seemed everyone had good faith efforts during the meeting.

Based on some situations from the meeting, I had to divert my efforts to getting other applications working. I had to fix the issues the group had with the emulator, so I spent a night researching qemu. I found out how to correctly use the emulator for our efforts. I posted the tutorial onto the discord. I also found out the limitations of the emulator. This helped in the first integration test meeting.

The group practiced an integration test to verify everything was ready. It was an hour and thirty minutes long. The group weighed through the possibilities of how to run the different programs on the raspberry pi. Additionally, the Pi had some problems connecting to the internet. This meant manually transferring the files via USB from our laptops. This did not work in some cases, because the package manager on Linux was required to install dependencies within my code. Thankfully, Max took the time to configure the Pi on his own later. Additionally the group saw the rpg maker game would not work on the Raspberry Pi.

Due to problems seen during integration testing, I had to write extensive amount of documentation for Maxwell. I also helped lead the group in deciding the new “architecture” for the game. Deciding between PyGame and HTML5,
Haofeng expressed his difficulty with using Python. I pushed for HTML5 with JavaScript to allow Haofeng an easier time. I also helped them with a singular prototype, simply moving a sprite with wasd keys.

In conclusion, most of my activities this week focused on damage control. I lead and mitigated through issues to provide my team opportunity. I feel exhausted from all of the deliberation, prototyping, and research done over the past week. I am proud of the accomplished activities this week, but I hope to work more on pure software development in the future.

**Problems**

The first problem I had to solve was my graphics. This seemed to be a reoccurring issue. I would develop and test in personal computer environments. I would then find out, porting the code to the Raspberry Pi would be an additional 10 steps. My client complained that I was not using an emulator. The emulator is very slow and really prevents on-system development. Typing into the emulator takes seconds rather than microseconds. The emulator also does not provide emulation of the graphics card. This means I cannot test anything graphically on anything aside from the Raspberry Pi itself. This lead to decisions as a team to avoid Graphical Layers within game development. This way others could emulate the game, even if they cannot emulate the dashboard.

The second problem was the client opting out of the chosen game architecture. The game was created and chosen by Haofeng. I did not have a lot of information about "RPG Maker." I assumed Haofeng had researched and considered options in creating the game. It turns out the development framework has legal issues in use. There is a royalty fee. Additionally, it does not port well to the Raspberry Pi. This means the team was forced to change game engines a week before project alpha completion. I helped mitigate this issue. We are now using JavaScript which opts for rapid development. It also uses similar graphics APIs to many multimedia libraries. This way the development environment is simple and effective. It ports over to the Raspberry Pi well. I did create a prototype, but it seems the team, mainly Cory and Haofeng, dislike the concept. It does not use an additional graphical layer or library, meaning some systems will need reinvention (I am going for the phrase "reinventing the wheel"). Whether or not Cory and Haofeng use my code or not, they will have to come up with something by next Friday.

**Plans**

I plan to have a working Game and Dashboard by next week. Wow! That’s a lot to promise. I am very sure of this though. I will code very very very very very minimally. I will follow the principle of simplicity rather than completeness. I want to see a successful project on the pedestal to make the client happy. The game and dashboard can be simple. Once a minimal viable product surfaces, the group can work on extra features later on. I cleared my schedule for most of the weekend. I plan to do something similar for the next week. I will crunch a bit. I realize this may invoke some poor feelings in some. I remember the TA reminding me to consider whether the grade is "worth it." This is more than the grade. I have high hopes for the project. I want to see it succeed.

**Blog Post**

- January 10, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz
Progress
This week was about reconnecting and getting ready to start development. Sadly I did not get to work on the project over break. That turned out fine though. My colleagues did not care. I got to hang out with my family. I am happy that things turned out well.

I also reached out to the professors, TAs, and clients. The client met with us on Wednesday. Mainly we went over what expo would look like. We also confirmed previously agreed upon decisions, such as custom sweatshirts and due dates.

I also started the GitHub repository. You will have to email me for access. I am really happy to finally have it up. I will probably continue working on it tonight. I really like seeing the work start to get completed. Currently the documentation for plans is up. I hope to get some tools up next before any code is put up.

Problems
My current problem is lack of enthusiasm. I think Cory is planning to re-brainstorm the game. I can live with this, but I want them to do the work in this case. I put a lot of work into brainstorming and planning the game. Ideas are worth pennies. I am not set on the Maze Of Dreamz idea. *However* the team changing concepts and the team actually doing another concept are two separate ideas. I have my doubts that they will actually start the new idea. Overall, I just hope to stay positive as development starts to impede.

Plans
I will try to do as much as I can with the DreamZDash. I have a plan set in stone. I think that with the amount of planning done, I can accomplish this project in a little over a month. I will be busy this month with interviews and job stuff. I do want to try my best. I have high hopes in myself. I hope to have my team here with me later down the line.

Blog Post
• January 17, 2020
• Tristan Hilbert
• Project: DreamZBox 2.0
• Team: CS29 Dreamz Catcherz

Progress
This week I was really focused on my club’s production. I also worked on some code for the project. I am working on the utilities portion of the code. These will be used throughout the project. It’s hard getting the first few modules set out. Making mistakes early on will make things harder in the long run. I have implemented doxygen and CMake with some difficulty. I think things turned out well. I will need to put more hope in the week 2 sprint.

Problems
I didn’t have a lot of time to work on the project. I think this was my biggest problem. Other classes have really hiked up the assignments. Hopefully, once schedules iron out, I can help out more. I also wonder when the other members will start their ends of the project.
Plans
I plan to finish the Kernel and Utility modules by the end of this week. I also plan to do an on-site interview. I think this week will have some good come out of it. I do think the work will get done. I need to get ahead to so I can help the other group.

Blog Post
- January 24, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress
When I submitted my progress report last week I had high hopes. I had not looked at my itinerary for the interview. I had not looked at the projects needed to prep for my job. Nor, did I look at the homework I would need to complete over the weekend. I did not complete any of the objectives I had for this week. I did participate in the meetings. I divulged my failure to meet requirements during the meetings. I also helped discuss implementation amongst my peers. I feel I tried my best given the conditions of my schedule.

Problems
I got really sick over the week. This made me feel really fatigue and put me behind schedule in many of my classes. I think after this week I will get better. I sadly took every spare moment I had to nap and nurse myself back to health. Because of this, and the other obstacles, I did not get anything done. This may leave features undone later on. There is no time to lose.

I will say I dislike having very little done in my code. It shows very little of my participation in this class. It has been hard this term to split myself up in such a way so all my classes gain a good amount of work periods each. I tend to work till completion on every assignment, procrastinating the ones with farther due dates. This will not work for DreamZDash, where the software still needs to be architected. I hope to solve this in future weeks.

Plans
I currently plan to finish the DreamZKernel, Hardware Status, and get something displayed. I won’t be able to test on a raspi, but I should be able to test on the emulator. I had Cory borrow my rasberry pi to try out QT Creator. I am still unsure if this will be used for the entire project or not. I wonder when we will have a feature freeze as to what will be in and out of the game.

Blog Post
- January 31, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz
Progress
This week I progressed really well. I did not finish everything I set out to do, but I have ample confidence the project will finish well this week. I was able to both finish coding the main utilities for the dashboard. I also got a Virtual Machine running for testing. The Virtual Machine does not act well to perfectly mimic the abilities of the raspberry Pi, but it provides the proper Operating System. I am using the x86 virtual machine, because it works faster. I kept running into issues by testing my code on a Windows based Machine. By switching to a Linux based machine I got to write accurate testing code. I had access to the POSIX based libraries put forth in Linux.

I will be using the virtual machine to practice my graphics code as well. I have decided to use SFML because it was easy to set up. It has packages built in the operating system which we can also install on the Raspberry Pi. I am really excited to have decided to add graphics. It is much harder to explain the success over a past week without showing something pretty on the screen. Much of the code written so far is just simple CLI applications. The Display object actually represents something I can demo.

Problems
I had a couple of problems this week. I first did not have my demo ready for the presentation. The main problem was the concept of the Virtual Machine. I had not considered using it, and I did not have the emulator ready either. I could not accurately test my code. I chose the Virtual Machine mainly out of speed-of-setup. It was really fast to set-up, but I also had to set-up SFML and compile for the first time with it. Due to these factors I did not get a successful compilation until 2 hours after the TA meeting.

Second, one of the group members is starting to spend too much time on their aspect of the project. Moreso, their graphics implementation is not halting pieces of the project. I needed two individuals to work on the game, but Cory is spending too much time trying to set-up QT Creator. I don’t want the Alpha to only comprise of Haofeng’s work. I suggested to Cory that he try to look at other aspects instead of QT. He said he would, but I have not heard anything since.

Plans
I plan to have a working Dashboard by next Thursday. This dashboard will launch an independent application and work based on inputs provided from the keyboard. The keyboard “wasd” keys will move the selector and “space” will launch the game. I think this will be doable in the coming week.

Blog Post
• February 14, 2020
• Tristan Hilbert
• Project: DreamZBox 2.0
• Team: CS29 Dreamz Catcherz

Progress
I made a lot of progress on the Dashboard. I did not get as much done as I wanted to. I added some animation and the main Game Selector object (I nicknamed it Game Preview). I still have not attached input nor any back-end applications. Turns out I accepted a lot of work with the different components I designed. I also have started conceiving of certain
issues and other features. I had to solve a problem of loading in textures and fonts. It turns out that they have to load in separately alongside the rectangle shapes. I would have rathered a utility class that contains both in the same constructor and destructor. Now I have to have two variables per “sprite” which becomes messy in the code. I wish I had spent more time trying to design alongside the graphics frameworks to prevent these problems.

The following are two lists of features. The first is necessary that I will have to bring in after Alpha. The second are Quality of life updates which probably will not make it in, but would have made the application work better.

**Necessary Features**

- Pop-Up Menus
- Pop-Up Usable Keyboard
- File ID System for Games

**Quality Of Life Features**

- Asynchronous Rendering System (Rendering Rectangles and Animating them on different threads)
- Assets Container System (Load in Graphics into one container and just use smart pointers)
- Database System/Shop API for Video Games (Something programmed quickly in NodeJS)

While it would be cool to implement these. I will have to ration out to what I can do. I am glad that I am staying ahead of issues, but some things are better left in hindsight. I will try to mitigate as many issues as I can, but I will probably leave a lot undone. I do not need this to be the next Android JVM system. I just need a dashboard that can launch games. I have not been able to make any progress on the game though.

**Problems**

I experienced a couple big problems this week. One blocked me from working. The other just ticked me off.

**Blocking Work**

I realize that work and emotions do not mix well. However, when I had a cancelled party this past Saturday I was hurt. I was really amping up for an amazing double date at this small gathering. I honestly had not dated anyone in a while. When the event got cancelled I got really depressed. For that reason, I took the weekend and spent my time away rather than working on the project.

Additionally, I had a Mobile Dev group design meeting. I had to put a lot of time and effort into completing the proposal. It should have been a 30 minute assignment, but the group butted heads a bit. In the end we solidified on the idea. I think it will turn out really well. I am slightly worried how this will affect my work for Capstone in the future.

**Meeting Professionalism**

I am going to speak a bit freely in this subsection. I will definitely have some poor spelling and grammar. I will berate the language. I understand that the group comes first. I really enjoy working on this project with the Dreamz Catcherz. I do expect some decorum while we work through this project. I want everyone to show ample effort. This means giving credit where credit is due. I would like “thank you” and “good work.” Needless to say my work on last week’s game was brought up without my permission nor consent. It was unplanned and done in lieu of a lack of work. My code was
actually shown twice without proper attribution. That really angers me. I am okay working hard for the group. I am okay with no praise in doing it (although my team constantly praises me). I am NOT OKAY with almost blatant plagiarism. If you are going to show off my work in a meeting, you better have a good explanation why. I wanted to bring questions like, “Where is your demo.” “How does this program work?” “Why did you choose this implementation.” Not only was my work shown without my consent, but it was also criticized. If you don’t like something I did, then suggest an alternative. Do not call it “wonky” and “hard to work with,” because I use something elegant and advanced. I can take feedback, but do your freakin work.

Plans
Due to the lack of time and the need to be finished, I will be switching into Python shortly to finish the back-end of the dashboard. I will also be finishing the front-end graphics implementation in C++. I am almost finished with that anyways. I also may expose the Kernel in C++ aswell since I know how to do it better with that system. With less blocking me this weekend I should get more done. Honestly, with the late transgressions held during development, I just want this project done.

Blog Post

- February 21, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress
I totally lost track of time and forgot to submit this report. I will write it anyways. The last week was pretty rough. I have now been maximizing my time working rather than sleeping. I will probably make up for it by the end of term. In any case, I finished the functional parts of the Dashboard. I did not get time to finish the back-end stuff in Python like I wanted to, but I was able to complete the front-end. There are a few bugs I will need to decipher later on, but it functionally works. I am really happy to see such an big work come together.

I also help put together the presentation for the Alpha Design Demos. I got a template from Oregon State’s online resources. I then made it editable for my group-mates. It seems I was the only one with something prepared. My group mates did not know what to present about. I guess they want to do a large demo. I think this is a missed opportunity on reviewing through code and showing the cool bits. In any case, I am looking forward to the upcoming thursday.

Problems
I had to get through a ton of assignments and projects for other classes this week. I also had some drama amongst friends. Both situations brought about some tribulation. I really need time to sit down and work, but life keep getting in the way. I have not gotten the ability to give a weekend to doing school work. I hope to mitigate this in the future. I may have to get more focused when working and try to allocate my time more wisely. It’s difficult to do so much at once. I am already missing assignments for my Defense Against the Dark Arts class. I do not think it will turn out that badly though. I am hopeful things will work out eventually.
Plans

I will hold on development for the DreamZDash and start contributing to the Maze of Dreamz. I hope to contribute significantly to the gameplay mechanics, allowing players to actually play a worthwhile game. Then next week we can feel for innovation, trying to finish out all the requirements by the given due date. This will be a long haul, but I am cool with crunching for the time being.

Blog Post

- February 28, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz

Progress

I made little technical process on the project this week. I started development on a forking process within the DreamZKernel. I want to try and get the State Machine right within this object. I think I can accomplish this before the due date of the beta. Currently the dashboard stay open when running the game. I want to create an alternative process such that I can close DreamZDash without having to worry about losing track of the game. I now believe this to be a necessary feature, otherwise the console will look rather jenky. I do not want to waste too much time on it, but it does require some IPC. This is one of the hardest concepts within the project. I have yet to accomplish this on my own.

I also worked through design review, making sure to produce enough work, content, and organization for the group. I recorded our group’s presentation to help with the video. I also provided the shared drive for the group to contribute to. I made sure everyone could speak, share, and provide to the presentation. Everyone did exceptionally! I was surprised more individuals did not ask questions during the presentations. I think having them early definitely kept people tired. In any case, it turned out well. I really enjoyed the experience. I am excited to see the feedback we receive.

Problems

I almost missed the design review completely. My alarm did not go off and I woke up 15 minutes before the start. I ran really fast and thankfully made it to the review. I was tired for the rest of the day. I should have set more alarms I guess. I am grateful to not have missed the meeting.

I also mitigated an issue in communication with Cory. Him and I had a great one-on-one conversation. I now feel more confident in making decisions towards the project. I think I overanalyzed what he was saying during meetings. This definitely removed a rather large stumbling block for me. I think it will lead some immense progress in the project. I am ready to get this done.

Plans

I hope to have a playable game by Monday. This will need to be slightly playtested by Tuesday. I know this whole term I have over-planned. This Friday I am planning to stay the whole day at the library. I want to work as hard as I can. This is crunch time. This is my opportunity. I have a playlist ready, and I am not about to fail.
Progress

I spent most of the week defining an actual architecture for the game/engine. I am using Entity Component System. It definitely took me a while, but now I can freely hotswap different game modules on the fly before compile-time. I also implemented some example modules, enemy with AI for example. This simply just provides movement.

Eventually I want to attach this to Dijkstra’s algorithm or some path-finding AI. I heard Cory talking about adaptive AI. I cannot program that. If Cory or Haofeng want to, then they can go ahead. I am already pretty booked. I need the following done next week:

Problems

I missed this week’s meeting. I was really rushed this week with other assignments. I wanted to make it, but my head was too deep in homework. I will make future meetings with Vishnu. I hope to surprise the group with some kind of deserts for my previous absence.

I also had to explain the new system to the others. It turns out, I may know more advanced concepts in Javascript than I realized. I promise I am not trying to complicate the code on purpose. Yet, the other individuals on the team already find my code convoluted. I hope to remedy this issue in the future.

Plans

The game is almost playable with a few missing pieces. I have seperated my work into a todo list of modules, that I call “Systems.”

- Lifetime (Objects which die after a few frames.)
- Render (A rendering system which prevents objects out of frame to not render)
- Spawn (A system which allocates spawn points based on player vicinity)
- GameStats: (Any Dynamic Variable changing i.e. score, mana, life)
- Life (At minimum)
- Menu (Something shows up when you “pause the game”)

Blog Post

- March 06, 2020
- Tristan Hilbert
- Project: DreamZBox 2.0
- Team: CS29 Dreamz Catcherz
**Progress**

This week I programmed Lifetimes and completed some generic functions for continuing the ECS-like architecture of the game. I met with the client to try and dignify a need to change the requirements in light of the April 15th code-freeze. These changes were minimal in outcome, but they required immense brain work to procure. I really felt like I could have done more. I am suppose to create the image file to finish off the night and induce the tagged version for beta. I really do not feel like the progress I have done is enough. I really want to accomplish more with the collision, allowing enemy deaths within the game. I also want to provide the game rules to follow given spawning for the player.

**Problems**

Many of the code changes my partners produced did not follow the ECS model. I tried to explain it well, but in general their code featured many hard-coded values. I think it would have been nice to implement a code review for the students to be allowed some insight into what could be done. One student wrote a small function, the other wrote 8 functions. I have already written 18. I am unsure how the others want the project to proceed. I think their is some difficulty in knowledge levels of JavaScript. I think I am choosing to use things for efficiency rather than understandability, but it is causing lack of inspiration for the rest of the team.

The Coronavirus has been a huge pain. No one is meeting up anymore, and many of the students have given up on the project. I do admit; I fear the cancellation of the expo. This would totally nullify the work put into these projects!

**Plans**

I will provide the following in the nearby hours.

- An image file of my current OS
- Source code for the current repositories

Over spring break I will continue to work on the following:

- Different types of enemies
- A finishing mark
- Menus
- Modes

I want to make the game seem more industrial without much of the work. The ECS framework will help in keeping my additions in check and repeatable with little work.

9 **Weekly Blog Posts – Haofeng Tian**

**Blog Post**

- October 18th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

Met with client and TA, discussed about the project and requirement.
Problems
The EECS group’s design about the game controller didn’t as same as what we imagined after the first meeting, so we need to make some change about the game design.

Plans
Complete the requirement document, study the samples that provided by the client.

Blog Post
- October 25th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
Met with clients and ECE group, discussed about requirement and progress for different groups.

Problems
The first requirement document needs to make some change based on the clients’ feedback.

Plans
Getting familiar with programming environment, study the samples.

Blog Post
- November 1st, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
Met with the TA, discussed about the project. Learned some samples about game design.

Problems
Lack of experience about designing games, some information is new to me. Thanks to Tristan, he usually posted the ideas and technologies he had on discord, so I can get start easier.

Plans
Complete the technology review, getting familiar with programming environment.

Blog Post
- November 8th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz
**Progress**
Met with the clients on Wednesday, talked about the current progress. Doing the tech review and researched about the methods for Game engine, Music generation and Image loading. On Thursday, met with the group and discussed the plans for game design.

**Problems**
The design of controller is changed by EECS group, need to make adjust about problem statement.

**Plans**
Going to start to do design gallery, then read the feedback about the tech review and do more research about the information on tech review.

**Blog Post**
- November 15th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**
Complete the tech review, and the first draft of design document.

**Problems**
There are 2 templates of design document are available for us, we need to discuss more about with one should be used.

**Plans**
Complete the final design document and prepare for the game making.

**Blog Post**
- November 22th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**
Complete the design document, make the changes based on the feedback form our TA.

**Problems**
The first draft has some redundant sections and need to add a relation diagram at the end of the draft. These problems had been fixed in our document.

**Plans**
Research more about the tools and technologies we need, practice with some coding skills.
Blog Post

- November 29th, 2019
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

Due to Thanks giving holidays, no class or meeting this week, analyzing materials we have searched at home.

Problems

No problems were found in this week.

Plans

Research more about the tools and technologies we need, practice with some coding skills. Also need to get prepared for final week.

Blog Post

- January 11th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

This is the first week of the winter term, we have rescheduled the meeting time with TA and client. After doing the in-class activity, we realized our shortage. As for me, I need to become more active and try to contribute more to the team in this term.

Problems

In the first week, all team members look good, no problem was found.

Plans

We scheduled weekly team work time, and created github repository, hope that we can get our game be playable as soon as possible.

Blog Post

- January 17th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz
**Progress**
We had a team brainstorm on Tuesday, add some new features to our game, and set the individual goals for coming weeks. I’m currently trying to make some small games by RPG maker.

**Problems**
It’s hard to find a good simulator for raspberry pi, Tristan helped me fix this issue.

**Plans**
Make a simple game by RPG maker and try to run it on raspberry pi.

**Blog Post**
- January 24th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**
Made a basic game by RPG maker, had a meeting with TA on Thursday and planed to add features to the game by next week.

**Problems**
Had problem when try to transfer files between emulator and my laptop.

**Plans**
Learn about how to use script to add more features to our game, need to do research about the language I have to use, and apply the real-time battle by next week.

**Blog Post**
- February 7th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**
Alpha demo looks work well during the test, discussed the plan about the next week.

**Problems**
The rpg maker is not a open source game engine, the client ask us to find a new plan to make our game

**Plans**
Make the new game using CreateJS library, learn about the new engine and try to make some small test game by next week.
Blog Post

- February 14th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

Have some images shown on canvas, test with some demo samples.

Problems

Had problems when put characters on the map.

Plans

Keep building the new game with JavaScript.

Blog Post

- February 21th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

Test with collision function in game, prepare for next week’s design review

Problems

It’s easy to test with 2-directional movement, but I need to make changes because our game have 4 directions.

Plans

Get ready for design review, make sure the speaking time is suitable.

Blog Post

- February 28th, 2020
- Haofeng Tian
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

Collision function for game is working with demo code, try to combine the work with map generation function that is written by Cory.

Problems

Need to fix bugs that was found after combining the function.
Plans

Start to write the basic attack function.

Blog Post

• March 6th, 2020
• Haofeng Tian
• Project: DreamZBox 2.0
• Team: Dreamz Catcherz

Progress

Completed design review, applied collision function to game play.

Problems

The format for game code need to be written in ECE style

Plans

Keep working on attack function, change the style to write codes.

Blog Post

• March 13th, 2020
• Haofeng Tian
• Project: DreamZBox 2.0
• Team: Dreamz Catcherz

Progress

Made video for report.

Problems

Since this is the last week of winter term, no problem was found this week.

Plans

Plans to add features to our game, need to complete the end of term report.

10 Weekly Blog Posts – Maxwell Evdemon

Blog Post

• October 18th, 2019
• Maxwell Evdemon
• Project: DreamZBox 2.0
• Team: Dreamz Catcherz
Progress
I attend the ECE and CS group meeting where we discussed the controller design, I was also able to understand more about what I should be contributing to the project as the primary firmware connection for CS and ECE groups. I was able to get a better understanding of my groups viewpoints to what they think it important, while looking over my groups individual problem statements. I was also able to think more about how our user interface should function, past the basic options. In a group progress sense, a meeting for the CS side we were able to come up with an official game design/style our group would be using. This would be a top down game style that would be procedurally generated. I found that most progress this week tended to be more group progress than individual. I was also was able to learn more about the accelerometer on my own time and think about how it could be implemented.

Problems
The first problem encounter this week was the controller design the ECE side came up with. Since this design was a single joystick our original idea of making a 3D/pseudo 3D game was no longer possible. This means we had to come up with a new game design, and the current game design is still shaky at best, as we only really have loose ideas for the current game. Another problem I felt we had was that I thought our requirements were a little open ended, which is something I plan on asking the client about.

Plans
The primary plan going forward for me personally is to start planning out game and making sure that it is even possible. This requires me to start learning more about the language it will be coded in and more about the hardware that will run the game. I also want to start planning the user interface as that is something we have yet to bring up during our meetings with as a CS group and our weekly with the ECE group. This week happens to be our bi-weekly meeting with the client, where I plan to ask more about our requirements for our final client requirement document. During this client meeting I also plan to ask about our game idea and present it to the client along with my CS group members. This way we can get confirmation on our game style from the client directly.

Blog Post
- October 25th, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
This week we were able to meet with a client and in that meeting we were able confirm the controller design with her and were able to run our game design past her. The client seemed to like our idea, or at least approve of it. The client would then ask what was out unique feature that would set it apart from other games like it. Since we did not have a proper answer for that question, they then asked us to have the idea ready by the next bi-weekly client meeting. We would later come with a plan that we hope to be seen as something to make our game stand out. This would be procedurally generated puzzles along with the procedurally generated maps. This is to try and keep an overarching consistent theme while making use of more procedurally generated ideas. After the group meeting, we individually
starting to think up user interface ideas. As the user interface is our other major part in the project and the client would most likely want to see our ideas by at least the next client meeting.

Problems
The issue we have is that we are still unable to confirm whether or not the game we want to make is actually possible. This will be more apparent as we look more into software and the hardware through research, this is one of the reasons why we cannot really solidify our plans/requirements any further at this point in time.

Plans
Since our client meeting are bi-weekly we hope to have our innovative game mechanic and our rough draft idea by the time the next client meeting occurs. As mentioned in the progress part of this weekly report we have an idea and now we want to solidify that plan and expand upon it until we think we have something truly unique to our game. Once again, we want to have this unique game feature done by the time of the next bi-weekly meeting with the client, this way we can get their input. Something we also need to decide on is if we will be using some open source tools to help us with creating project or attempt to make our own unique tools from scratch for the game. The last plan I currently have it to look more into the software and hardware aspects of the project, this way we can know our bounds to make sure we can create the project we want.

Blog Post
- November 1st, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
This week during the meeting on Wednesday between the ECE group and our group we talked more about our ideas between the group, this includes a new controller design. Personally, I was able to learn more about my role in a firmware connection between the controller and the software this is how the connection will work. I learned that this will include usart and i2c connections. This week we were also able to pitch our idea to the ECE group, but it will most likely go through some changes as the controller seeing changed so we can implement new ideas. I was also able to discuss more ideas with another group member on user interface designs. Thanks to the research project drafts we were also able to start thinking of more practical ideas for the project and think of all the different steps we would need to implement into our project.

Problems
A problem we had this week was the fact that the controller changed meaning we have to think on new ideas. This includes a new scroll wheel which we will have to implement in the game. Luckily me and another group member have already talked about its inclusion and plan to pitch it to our other members in the next meeting. Unfortunately, not all members attend the weekly Wednesday meeting, meaning two of them still don’t know about the specific design of the new controller and have seen nothing of the rough draft ideas we have for the user interface. This has also hindered talk about the user interface which is something we still don’t have much done on and need to work on before next Wednesday which is out client meeting.
Plans
I want to personally know more about the firmware topics so I can discuss more specifics with the ECE team next
Wednesday as well. Then finally I want have a group sit down where we discuss more species within the group as it
feels like some of the group is not caught up, and we want to get our unique feature for the game figure out along with
the new controller design. I also want to meet with our group after we have done our research this way we can learned
more about what everyone thinks their part is and how we as a group can move forward on learning more about our
individual parts and any other parts if necessary. I would also like to contact my group this weekend to see what exactly
they are researching as well, this way we can make sure everyone’s role is clearly known to them.

Blog Post

- November 8th, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
This week was a one of our client bi-weekly meetings. This week we were able to meet with the client and ECE group
once again on Wednesday. During the meeting both groups were given an extension to our standout feature. However,
we still told them our current plan. That planned unique feature being, was our procedurally generated puzzles for
each run of the game. This way the game continues to be unique on every playthrough rather than having unique maps
with eh same puzzles on each run through of the game. The client seemed to think the idea was fine, but as a group we
would like to improve on the idea to perhaps make it more unique.Individually we were able to discuss our research
on the project, I ended up dealing with I2C connections, USART connections and HDMI functionally as my role was
firmware connections. We were able to properly meet as a group this week and fully explain the new controller design,
provided by the ECE team, to each of team members that may still be unsure on the idea of the controller. During that
meeting we were also able to make more progress on figuring out core game mechanics we want to implement. For
labor division this week it mainly individual group work on the review process, but Tristan also brought some more
ideas to the table for the video game in terms of controls and abilities we could use.

Problems
One of main problems is the fact that we still do not have any set-in stone languages for actually doing the assignment.
That is what the research was primarily for, but at the moment there is no real talk on how it can be used to accomplish
the goal in a serious environment. This builds on one of the problems of last week, even though the research is done,
we are still coming up with ideas that may or may not be possible with our current information. This may requirement
more in depth research. We also need to think of more ways to make our game truly unique as by next meeting we
would like to have something more than just procedurally generated puzzles. Finally, we were unable to show our client
our decided-on user interface mockups. Due to that we will need to potentially wait two more weeks until the next
client meeting or send it to them over email as PDFs.
Plants
Going forward the first thing to do is to look more into actually doing the project on top of the research we have done. This would be more side things such as graphics and potential music we could use for the game. This is something I will bring up in our next meeting as both of those are important to video games. I would also like to at least show our client user interface idea as I think that is an integral part of the project and one of the things, they specifically requested of us. I would also like to look more into improving our unique idea, as they effectively gave us another two weeks to improve on our current idea as the ECE team was not ready. The new controller design allows for plenty of new features we could implement in the game as well, which is something I would like to discuss with my teammates in an upcoming meeting. The next major thing is the design document which we should be working on as group this weekend and Monday. It may help clarify some design questions we have.

Blog Post
- November 15th, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
As a group we got to discuss more ideas about the projects design, which would then be applied during our project design first draft. We got to view each other’s tech reports so we could see what everyone researched specifically. The group meeting this week was canceled so I could not bring up my ideas up to the whole ECE group. Since I am dealing with firmware this puts me back a bit progress wise, even though I spent more time looking into the libraries and getting more familiar with them. In terms of group project, we really did not have any full group discussion this week in person, but we were able to discuss more ideas online. We were also able to get the design rough draft done along with some state charts, even if was at the last second. Tristan, Cory, and Haofeng did a good job helping with the design document.

Problems
There are a few problems this week which involves the primarily the draft design document and the user interface. For the user interface, unfortunately since the group weekly Wednesday meeting was canceled there was not a really good time to bring it up as a full group. Another problem is the design document, it was mainly due to the fact of how confused we all were when working on it, this is less an issue of the document itself but a lack communication/planning. This may just be a one-week problem as we have now all talked it out, and this has never happened before this. The last issue is not necessarily an issue, but rather something I would like to improve on. It is that our unique feature for the game is actually due next week during the meeting and we have yet to really discuss our current feature and try improving on it. While the client seems to like the current feature, the feature itself is not entirely unique which is why I am worried and would like to discuss it more with the group. However, like I said the client seems to like the current idea of randomly generated puzzles so it may not be a real problem. I just want to expand on the idea if possible.
**Plans**

Going forward we definitely should plan to keep more open communication on the project documents from now on. This is something I personally should work on, as I am not the best at communicating at times. Something I also personally need to do is get in contact with the ECE group as a whole and ask them what they think they are handling for firmware. Then what I think I am handling for firmware. I do not want to make the mistake of having one group working on a part of it if the other group already plans to handle it. By contacting them I will also be able to get more information about the controller and how the Bluetooth connection works. This should allow me to get more of the jumps start on my part. I would like to then discuss our unique feature this upcoming Tuesday as that will be the next time, we should all be able to meet as a group. As I said next week is our next client meeting and they will be checking in on us it would be great if we could add to or unique feature. Another obvious plan is to improve our design document. I would also like to bring up our design document to our client and get their input on our ideas. This is mainly due to the fact that our client was the one who created the original project and turning to them for recommendations would help us in the long run.

**Blog Post**

- November 22th, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

This week was one of our biweekly client meetings. At this meeting we got to discuss the design document with our client and some of our thoughts and progress on it. We were also able to get some questions asked directly to the client. For example, this involves the video we need to implement in the DreamZBox 2.0’s start up screen, which essentially serves as a pseudo advertisement for the console itself. Another question that go answered was the fact on whether or not we need to keep the dashboard running in the background during the game. For this the client gave us our standards, which is that we would not have to keep it running as long as it did not take too long to load the game dashboard again. Personally, I made some good progress this week. This week I was able to talk more with the client on what they expect out of my firmware part and assigned me a section they would like to see done with the firmware. I was also able to talk more about my side of the project to the ECE side and was able to at least better plan who is doing what among the ECE and CS firmware. We were also able to discuss the final parts of the design document. In terms of the unique feature I no longer believe to be an issue as the client seems content with what we currently have.

**Problems**

Luckily comparatively to last week the problems were not too bad. This week we no longer struggled with the design document, mainly thanks to Tristan’s hard work talking to the TA for requirements and on the paper itself. I really like the design of the dashboard that we have for our group, but it is definitely something I feel like we should bring up the client as always for approval to be sure. This might be me being paranoid, but it is still something I would like to see happen. The reason is didn’t happen this week was due to none of the meeting attending group members having the actual physical design of the dashboard. I would say we could bring it up in the next Wednesday meeting, but
unfortunately it is canceled due to known absence for the holiday. However, it is not the client meeting so in that regard we should still be fine as that is bi-weekly. The final issue is that of the dashboard suspended in the background. When talking as group it seems the rules, they placed on us could not be met if we shut the dashboard down completely, so we must think of a way to keep it suspended in the background during the game.

**Plans**

Going forward the first thing to do is to look more into the background running process. I personally would like to get our final design document to the client along with our dashboard for approval. However, that mainly falls to design document as it is more important for the time being. Something I need to personally do it get started on my personal task, this is to work on Bluetooth connectivity between a basic controller and the raspberry pi while the actual controller is still being built. This should then be transferable to the actual controller design when completed. Something I would also personally like to do is once again have another group meeting on Tuesday that goes over each of roles and what we play to do in the upcoming weeks. This is to get some level of groundwork for starting each of our projects soon, as I believe we all at least agreed try and start during winter break. I will continue to look more into my topic of firmware connectivity in preparing for the upcoming client meetings and group meetings, from where I can hopefully give good input to try and assist my ECE firmware group member. Overall things are going much smoother in comparison to the previous week.

**Blog Post**

- November 29th, 2019
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

As mentioned in the last report, there was no real meetings for this week, within the CS team or as a whole. This week, while I did not have much time, spent most of my time on the project looking more into my part of the project along with the idea of keeping a background process running for the dashboard. Firmware side, I learned more about my part and now have begun to look into actually starting the progress soon. This is due to the fact I believe I now have enough information to start my implementation. I have been in some contact with the ECE team, so I do know what I specifically need to do on the firmware side. I will also soon hopefully have access to the DreamZBox 1.0’s original controller for some test connectivity

**Problems**

This week there was no real problems, or at least any major ones that I have heard of, just somethings I want to keep in mind. The only real issue is that we really didn’t communicate this week on what any of us were planning as the group meeting was canceled, and we did not really meet a CS group either. This means that the CS group will be going into next weeks client meeting, only having met once. While this not an issue really, since the next time we are meeting is Tuesday, with the meeting Wednesday, this may lead to us being a little less prepared than usual. Personally, I need to get my hands on some usable hardware so I can continue coding and working on my part in a timely manner. This
means I will need to get a working raspberry pi I could use to implement my firmware coding on. This is something I will need to procure myself in the upcoming weeks. A piece of hardware I will have to wait on, is the prototype of the DreamZBox 2.0’s controller which is something the ECE team will be handing off to me and the ECE team member that is also working on firmware aspects. Something we also need to get done is our client verification. This is something I believe would be good to do this upcoming Wednesday as we would already be meeting with the client.

**Plans**

The first and most important plan for me will be to bring up the idea of getting our client verification done during the client meeting, to my CS group when we meet next Tuesday. I would also like to tell them I plan to start some implementation as soon as I can acquire some proper hardware to test with. This primary is in reference to getting either the prototype to the DreamZBox 2.0’s controller. I would also like to ask the ECE team when they think they may have a prototype done by, at least if they have a rough idea. This way I can at least have something to show for the Bluetooth connectivity of the project’s firmware. I would also like to hear some other their thoughts on when they believe they will be able to start some implementation of their parts, along with any ideas they may want to bring up for the client meeting that Wednesday.

**Blog Post**

- January 11th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

This was the first week back after winter break, in terms of progress as a group not much was done. This due to us primary discuss our new meeting times, TA meeting times, and client meeting times as a group. This week was also a client meeting in which we had a chance to discuss with the client and ECE team, but is was more just talking about the term as an overview and what they expected out of our group. We would then talk about what we think we should do on the project going forwards into the coding/development section. This week the group GitHub for our projects code was also set up. Personally, I was able to get started on my Bluetooth connections. To do this I am using a raspberry pi with Bluetooth connectivity and then coding to it. The controller is not yet complete, and I have not been able to get a hold of the original controller yet which I will need for testing. Right now, I am getting more familiar with my part of the project. I know about basic connectivity, but I am still working on getting the whole thing properly coded. I am currently trying to get a connected to the pi with my phone for connection tests but so far, I have failed to get it working as expected. I think this may be due to failing to correctly set up my raspberry pi correctly.

**Problems**

Being the first week back from break there was not many problems this week. One thing I would like to discuss as a group is what exactly each group member is working on. At this point in time I think I have made it clear from my perspective that I am working on the Bluetooth connections between the raspberry pi and the controllers. Once I finish my section, I would like to help work on other aspects of the game as well, but I am currently far off from doing so.
The only other problem is a personal issue with the Bluetooth connectivity. The only issue is that I have failed to get it properly working at this time. I still have time to get it done as that controller is not ready to be tested and it is still week one of the term.

Plans
Going forward my plans are to simply get the Bluetooth controller connections done. For this I would like to get my phone connected to the raspberry pi using Bluetooth. Then I want to have my phone be able to send signals to the raspberry pi, in which it would perform some action noticing the connection message/signal. This way I would be able to show I have a connection between the raspberry pi and a Bluetooth object, that can send and received signal properly. If the controller is not yet done, I could then configure my phone to more accurately represent the controller. This way we could at least have something for testing the created game. I believe by accomplishing what I listed, it would then be easy to configure the code to then match the raspberry pi with the controller. I would like to be able to have this Bluetooth connection done within the next two weeks. This way I can show my progress to the ECE team early in the term and see what they have done with on the controller.

Blog Post

- January 17th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
This week was relatively quiet when it came to my part of the project. Last week I was trying to get my phone properly connected so I could begin to write code that would pick up sent signals. I was able to get Bluetooth working to recognize my phone. I am currently working on getting messages sent through my phone to raspberry pi over that Bluetooth connection. Then I can hopefully move onto sending actual signals for the game. This week I met with my group on Tuesday, in which we had a discussion about the parts of the project we are working on. During the meeting we also brought up some new game features we would like to implement, given the time constraints we have to work with. This week was also a TA meeting, in which we as a group would tell TA our individual parts we would be working on. The TA would also explain more about feature due dates such as needing alpha functionality testing by week 6. This term all ECE and client meeting are now every other week, so we did not meet with the ECE team or the client this week.

Problems
During this week there were not too many problems. The issue was that we are still not entirely sure on parts of our game design. We are also still experimenting with ideas surrounding the multiplayer aspect of the game. Overall, we have the core ideas down, like it being a top down game where the levels are randomly generated, but we are still unsure of some of the more in-depth features we would like to include. As a personal problem I am still waiting on the ECE team to give me a definitive date of when I can have access to the controller for testing purposes with my Bluetooth code. Similar to last week I have still yet to get the Bluetooth coding to fully work correctly at this time with signal sending, but this is not an issue yet as it will be due with alpha testing around week 6.
Plans
Going forward I want to try and get input reading done correctly by the end of next week and then done by the end of week 4 at the absolute latest. I also still need to research more about the code I plan on using. I would also like to contact my group for another possible meeting. In this meeting I would like to try and discuss some of our more in-depth feature ideas and their plausibility. During the next meeting I will also ask about the progress of the ECE’s created controller and when they think it will be done allowing me to test my code with it. Worst case I know they still have access to the old controller, and I will see if I can use it, if the ECE team does not currently need it.

Blog Post
- January 24th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress
This week was another TA meeting and we met with our client, we did not meet as a group this week. During this week I had a breakthrough, not only was I able to get my phone properly connected to my raspberry pi, I was able to send serial code over the raspberry pi. From there I was able to write some code that then allowed me to use the serial communication from my phone and have it received in a python script. In that script I would send specific messages which would be picked up by the script and it would respond with the correct phrase. From there I was able to get it to make inputs by importing in a library which would “push” keys. This could be used in the python script and it would then allow me to make movements in basic keyboard games. During the client/ECE team meeting I was able to get clarification on how I would be receiving my inputs. I learned that it would come in the form of a string from the actual controller, which I would then be able to decipher to the buttons being pressed on the remote. During the TA meeting we would each display something we had worked on to show a sign of progress. Unfortunately, there was no display port in the room so I could not show the pi working in action. Next week I will prepare a small video showing the Raspberry pi working properly with my serial inputs, this way I can show my team and the TA I have made progress.

Problems
During this week there was not too many problems. I am still unsure of when I am going to get the prototype controller. In this case, now that I have the Bluetooth serial working, so I would have no use of the old controller so for testing purposes. I still need to wait for the new design to be complete. While I now know the input type the controller will send, I still do not know the specifics on the string. This means I cannot begin working on accounting for those input stings. I plan on asking the ECE team more information about this in the future meetings. I am also at a crossroads in terms of my project, I can either being to work on having more in-depth serial controls or I can begin working on a multiplayer controller. There are also a few things I am unsure of now that I understand how Bluetooth works with the raspberry pie. This idea involves how the controller will connect to the raspberry pi.

Plans
I was able to get all my serial reading down by the end of week 3 which is great, now I will start working on getting multiple controllers/phones to connect. I also want to work on more in-depth controls from serial, as right now it would
only work if the game was keyboard controls. This is a way we could create the game, but I am unsure on how my team wants to implement the game. I also need to learn more about the Bluetooth controller the ECE team is making. I want to know what exactly the serial string will contain when sent to me, as that will allow me to start coding as if my phone is the controller. So, when the controller is ready for me to use there will not be any configuration issues. I also need to do more research about raspberry pi Bluetooth, as right now I do not think the controllers will automatically connect. This means I will have to write an additional script which will force the controllers to connect when paired.

Blog Post

- January 31st, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

This week was very successful for me as I was able to accomplish one of my goals. This week we also had a TA meeting, and next week will be our client and ECE team meeting. Last week I mentioned that I was able to get a part of my project done, as I could get my app to work with sample pi games. I ran into an issue while going through more extensive testing that needed to be fixed. Here I had to decide on whether or not to work on multiplayer or continue to try and debug this for the upcoming alpha functionality. I decided that I need to fix this bug before moving forward. For this I was able to use a virtual keyboard that would press keys based on what serial signal it receives like I mentioned last week. I eventually found the bug; it took some time to find a proper fix and then I began working on the next section of the code. This section of code would be the multiplayer section. I am still currently having issues with some parts of multiplayer, but in terms of alpha functionality it is pretty close to being done. As a CS group we were able to talk more about implementation of each part project. We eventually decided that next Tuesday we would meet a group and attempt to get all the projects working with one another, this way we can have all our groups functionality working together for the upcoming alpha testing. When meeting with the TA, I was able to show off the short video I made showcasing the work done.

Problems

This week I had an idea on how to handle multiplayer by I am now unsure on another part of my project. When thinking of actual consoles, they typical can sync controller automatically on start up as long as the controller is on. Like mentioned last week, I am not exactly sure on how to do this, but this is something that needs to be done going forward. I also still do not have the serial string that will be sent to me from the actual Bluetooth controller. This is something that has not yet been provided to me by the ECE team. I would like to understand the string being sent to me as then I can start to account for it. When it comes to internal issues in the group, there are none so far. At this point I still do not have the controller, but I have been told to expect it before the end of the term, which gives me an idea of when I will get it.

Plans

With the serial inputs of my phone to the sample pi game now working properly and I have some basic multiplayer controls working, I need to now start expanding on these ideas. I want to make sure joint controls work; this means
that if a player wants to attack and move at the same time then one of the controller inputs is not lost on the game side. For multiplayer I need to find a sample game to properly test it with, but I am pretty sure it is working as I can see the second players inputs working in terminal. I will need to find a way to get the controller to auto connect on startup of the console, this will probably be done by a script. I want to properly show off what I have done on the game to the client in the next meeting, specifically the serial inputs and multiplayer functioning correctly. As a group, if we have the whole project fully together for alpha functionality, I would like to also show that to the client if possible. I would also like to talk to my group members about the upcoming poster project, as it is something we will need to starting thinking about. I will ask the ECE team about the serial message they are sending to me, but I expect it to not yet be ready. However, this is fine as I still need to refine some parts of the project I am working on.

Blog Post

- February 7th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

Since this week was a client meeting, I was able to ask questions to the client. In particular if it was ok at this time to designate a controller the player one controller and the other controller as the player two controller. This is something the client was ok with, as this will help in the long run for testing but is something that should be avoided in the future if possible. The game controller now can connect on startup, as the python script will now run on rebooting the raspberry pi which was one of my goals to get done. Unfortunately, the python script still crashes when the connected device disconnects. I also wanted the controller to auto connect on startup, however the phone app was not useful for this as the pi refuses to connect to a device that is does not see as running. For testing I bought the Bluetooth part that is used in the controller and plan to try with it rather than the phone from now on. The python script still closes when it loses connection. As a group we were able to meet and then try to connect our parts together. For the most part this worked correctly, with the controller Bluetooth code working with the generated dashboard graphics. The only issue is the current created game in RPG maker, as it would appear upside down and was virtually unplayable due to the graphical issues. This is not an issue for the future, which will be explained in the problems section, but did show we were able to load the game onto the console. This week was also a client and TA meeting. In the client meeting we talked about our progress on the project and in the TA meeting we showed our progress on the raspberry pi.

Problems

This week there was one massive problem we encountered. This was at the client meeting, the client does not believe RPG maker to be a suitable programming software to create the game on. This has kind of sent the group into panic mode as alpha functionality is due at the end of week 6. However, for the sake of alpha functionality our client said that we could use the RPG maker game to show off in place of the final project. At this time the group members who are in charge of the game have found a new base library they can use to create the game. For my part specifically, I am still waiting on that serial string that the ECE will be sending to me. It is vital that I know it going forward and I am hoping to meet up with the specific ECE group member that is working on the string. I still do not have the prototype
controller, but now that I have the electronic part that deals with Bluetooth in the controller I can try and at least get the code the ECE team is working on for proper testing.

**Plans**

I want to get multiplayer properly working, but I need to make sure the ECE team member is ok on connecting the controller on the second Bluetooth port. This is something I plan on asking them in the next meeting. I really need the serial input sting so I can start perfecting my controller Bluetooth connections, this is something they did not have for me yet, but I will continue to push for this going forward. I need to get the electronic part up and running soon so I can begin proper testing of my part of the project. I want to have the controllers auto connect on start up of the console if they are powered on, this is not possible with the phone app, but I believe that it might be when using the electronic Bluetooth part. If not, I need to have discussion with the ECE team and client about how we are going to get the controller initially connected. I still need to make sure joint controller work as intended, that is if the user would like to move forward and attack at the same time. I need to write a shell type file that constantly tries to connect the controller if they miss the initial startup connection attempt. This assuming that it is possible to achieve the connection from start up of the raspberry pi. I finally need to make sure that I create a different shell or python script that constantly runs the Bluetooth scripts in the case of a disconnect.

**Blog Post**

- February 14th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

I was able to get the multiplayer working properly, and it was only to port one rather than an additional second port. This means that I will no longer have to ask the ECE side team for assistance on this matter. Something else I also accidentally figured out was how to prevent the python script from crashing when the client disconnects from the server. It was part of my solution for getting multiplayer to work, which involved threads. Specifically, multithreading was used to get multiple clients able to send to one port at a time. Also, now when the client disconnects from the server, that specific thread will close rather than the entire program, thus fixing my issue. At this point in time I am essentially completed my Alpha functionality. I was able to ask about the serial string to the ECE group member who in charge of it. I will soon have the code they are using directly on their electronic Bluetooth part. This means I can program the electronic part I have in order to test if the auto connect feature works. I already have the code written for the auto connect on start up to work, but as mentioned before it does not work properly with my phone serial app. This week there was a client meeting as all client meeting are now weekly rather than bi-weekly. This is because, as we get closer to the due date we can stay in closer communication with the client. At this TA meeting I was able to explain what I have done and what I will be doing going forward on the project. There was also a TA meeting, however due to a misstep on my part I ended up missing the meeting. Luckily for me my group members covered for me this week and gave the TA my updates.
Problems

In terms of problems I have only really one issue left, which is the controller auto connecting on start up. I have the code written for this and have tested it with other blue tooth devices, so I am hoping that it will work as intended. All I have to do is test set up the electronic part and integrate the given code from the ECE team. If it does not work, I will have to try and reconfigure the code, if it still does not work after that I will need to meet with the client and ECE team to figure out what happens next. Luckily, they have been providing some updates that makes it seem like they are on the right track. However, next time we meet I would like to see what they have done in person. I have offered to help with the game when my part is finished, as due to this unfortunate circumstance, I would like the original game we had in mind to still be completed. I still need the serial code string; however, I understand if it is not yet ready.

Plans

At this point my only goal real goals are to test the controllers to make sure do not interfere with one another and try to get the auto connect on start up working properly. For the controller interference I will write code that sends on two clients, both connected to the pi, at the exact same time. If both messages do not make it through that should be sign of some kind of interference between the two controllers. For the auto connect on startup I will attempt to use the code I currently have. If this fails, I will manipulate the code until it works, or I deem it not to be possible on server side. If this fails, I will need to inform the client at the next meeting as this is something, I was not sure of it being possible originally. This also goes for the controller interference. I also need to check multiple inputs at a time, which is another thing that will require me to hard code as tests. I will also ask the ECE team member in charge of the controller if I can see their code. This is so I can get a good idea on how it will affect my Bluetooth code. I also plan on working with my group on the design review in the upcoming weeks. I plan on having this all done hopefully by next Friday, that way I can focus on the design review. I will also make sure I attend the next TA meeting as well.

Blog Post

- February 21st, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Progress

I lost about half the weekdays due to being sick and did not really get any work done on the electronic part. This means I was unable to figure out anything on trying to get the controller to connect on start up this week. This week I spent a decent amount of time looking into the design review and thinking about what I am going to say. Along with creating the slides that I will be contributing for our group. I have been able to figure out some of the tests I am going to administer to my program and will be attempting to use them later this weekend. I was also finally given how the serial code string will be sent to me over the Bluetooth connection. We did have a client meeting this week, but it was more an informal thing where we took a group picture and gave small updates on what we currently have. I was able to see some of what my group had done for the game, and ask some questions, but I would still like to possibly help in the future. I was able to find a different serial app for the phone which functions more like a controller. From which I was able to test a few things, like if the code could handle a constantly flow of information. Luckily it could, which
was something I was worried about. There was also a TA meeting this week, in which we would do a demo of what we planned to present in the upcoming design review. Overall, I figured out how I should go about testing my code this week, I worked on my slides and information for the design review, and I am now able to start integration testing.

Problems

I still need to get auto connect on startup working. That is the raspberry pi should try and auto connect to the controller, when the raspberry pi turns on. Like I mentioned before I have some code that I believe will work, I just did not have enough time this week to implement it. If it fails, I will need to report to the client and ECE team to see how the client would like me to approach this going forward. As there are other solutions, they just may need the ECE team’s involvement. It is important that multiplayer works as intended, as if not then I will need to contact with the ECE team to tell them to connect one controller on port 2. This way there is no overlap and no chance of inputs being lost. As that is the primary concern right now, as I am connecting two clients to one connection server at the moment. I also need to make sure the raspberry pi is ready for the demo testing we will be doing next week during the design review. As right now it is still lacking some vital update code, this is something I will ask my group members to bring during our weekly team meeting.

Plans

Going forward I want to work on get the auto connection on start up. I have the code; I simply need to implement it with the electronic Bluetooth part. I will also need to get the updated game code from my team members for the design review. This way when I demo the code during the design review, it will properly show what we have accomplished for alpha functionality. I also now have what the serial code line will look like from the ECE team. Thanks to this I can implement that serial code line into my electronic Bluetooth part and implement it into my base code. This way the client code will be able to properly receive it and understand messages sent from the controller. This is something I would like to get done rather fast, as this way I can start to integrate with the ECE team directly. I am worried about integration more than anything, especially with the given time we have to do so. Which is something I plan on talking to my team about for the future. I also have some tests done specifically for testing interference between controllers. This is something I plan on testing as soon as possible. Also, with the new serial communication app I have on my phone I can send constant messages, doing this with two phones should also serve as a good interference test. I would like to have results by the next client meeting, which also happens to be a day before our design review. If I can have all of this tested and figure out how auto connect on startup works, I should be able to be done with my part within the next week. It mainly depends on if the auto connects on startup code works, and if not, how the client wants me to deal with the problem going forward. From there I would like to help the team creating the game if they require it.

Blog Post

- February 28th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz
Progress

During the client meeting we showed off some of our ideas on what we would be presenting during the design review and the progress we made from last week. For our TA meeting, it took place after the design review, so we would give a quick update on what happened during the design review. This was what went well and any possible mistakes that happened during the design review. Unfortunately, I ran into a few mishaps during the demo part of the design review, but I feel like the presentation section of my part went rather well. When it comes to progress on the project itself, I had productive week of testing. During some of our team meetings it was pointed out that multi button pressing was not in fact working as intended. I thought about it for a while and then came up with a solution. However, the solution does not work with the phone app’s that I am using to test with, this is due to limitation with the apps themselves. This solution will work with my integrated solution with the ECE team, as the created controller will not have the restrictions the phone app does. This is something I immediately checked for after running into the issue with the phone app, since I was worried it would not work either. I was also able to test more edge cases that I thought would be issues and thankfully none of them turned out to be actual issues. From there I was able to start my integration with the serial string that was given to me by the ECE team last week. For the most part that integration went well and gave me some insights on how to improve my code for the final part. I was also able to run more interference testing and so far, I have had no real issues.

Problems

I once again need to work on getting the auto connect on start up to work, this is something I would like to have working by the end of next week at the latest. I mentioned last time in my progress report that I thought I could have my entire part finished by the end of this week. Unfortunately, time is starting to become a real issue. Going back to the auto connect on start up, I really need to get working on the electronic part before I run out of time. It is the only way I am going to know if the process is possible or not. I also need to start thinking more about my shell code that will keep the controllers connected in case they disconnect. This shell code will reconnect the controllers in they somehow disconnect while the console is being used. I also need to run more interference testing. While I was able to confirm no interference between both controllers on some tests I ran, that does not mean that will be true for all possible tests. Lastly, I want to run more general tests on my code, as discovered by a group member multiple button presses did not work when I thought it did. This means there is a chance other issues exist that I do not know about.

Plans

Going forward I want to work more on testing and the auto controller connecting on start up. Like I mentioned before I have code that works with other Bluetooth devices, specifically my Bluetooth headphones, but I still have yet to test it with the electronic part. This is because I have yet to set up the electronic part. This is one of my goals going forward, I really want to get this done over the weekend, but it will require some help from the ECE team as it runs with their code. It is vital I get this done as soon as possible due to this part involving both the ECE team’s code and my own code. This is because if the auto controller connect is not possible, I will need to let the client and ECE team know. The ECE team has informed me that I will also need to send messages back to the controller to light up some LED lights on the controller. I have an idea on how to accomplish this, but this is still something I will need to do in the future. I also need to finish my interference testing for multiplayer. I was able to get multiplayer working without any interference but, that does not mean that interference is not there. I still have some back up tests I will be trying in the up coming weeks.
I also need to finish integrating with the serial string code the ECE team gave me. This is something I have started to do along with testing, but it is not yet finished at this time. I will also try to stay in close contact with my CS teammates who are working on the game directly. That way I can always be ready to test their newest versions of the game with my Bluetooth client code. I still only have a little bit of work left to do, so I would really like to get most of this work done by this weekend. At the very least I want to have most of this code done by next weeks client meeting, this way I can inform the client if the auto controller connect is an issue. The beta functionality and end of term reporting is not due for a little while, so I do not have to worry about them yet.

**Blog Post**

- March 6th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

This week was primarily just testing out my program for interference and general testing. This week we had a TA meeting and client meeting, unfortunately I had to miss the TA meeting due to accidentally scheduling a different meeting over top of that time. During the client meeting, the client asked me for an update, and I told them my worries about the auto connect on start up. They told me not to worry about getting it perfect and worst case the both teams would figure something out. I need the ECE team’s help with this auto connect on startup, and this weekend I will be meeting with them directly to get their code for the electronic Bluetooth part. Once that part is set up, I can move onto testing the electronic part with my code. Like I mentioned before if this code works, then I will have nothing to really worry about moving forward. As this auto connect is the last part of my project. In terms of integration, I have begun to write more of the integration code for the client part of the Bluetooth server. This integration code is something I plan to confirm this weekend when I meet with the ECE team, just to make sure nothing has changed during the time they gave me the serial code string. Also, for integration I have just received word that the controller is functioning properly for the ECE team. This means in the upcoming weeks, when I actually get the controller, I can begin rigorously testing the server code on the raspberry pi with the client code on the created Bluetooth controller. I have also almost finished my interference testing between two client devices. At this time all the tests I administered have passed, so I believe there is no interference at all. However, as we get close to actually testing with real controllers, I will probably re-test on that hardware just to make sure there is really no bugs with the system. I have also figured out how to send serial strings from the client back to the server, but I am still unsure on what needs to be sent back. As the ECE team requested that it was possible to send messages from the server to the client, in order to turn on certain LED lights on the controller.

**Problems**

My main problem still remains to be that the auto connection on start up is not yet complete. At least now there is some plan and timeline to finish the auto controller connect on start up. With that problem out of that way, or at least planned to be, this draws on the second part of the connection on start up issue. This issue is that if the controller disconnects while the raspberry pi is running, how will that client reconnect. My main idea is that I should have shell code that is constantly trying to connect to each of the Bluetooth MAC addresses. In theory if this works then it might actually void
the need for the auto connect on start up part of the code. However, at this time I believe this process requires testing, as I am unsure on how the raspberry pi code will react if it tries to connect to a device that is already connected. This may disconnect the device, mess up the Bluetooth connection for that device, or do nothing at all. This is something that needs to be tested going forwards and is the biggest concern for this part of the project. I need to make sure I am more aware of TA meetings, as this is the second time this term I have missed one due to being careless. I also believe I have not done enough general testing, like I mentioned last week there could still be bugs in my code that I have not yet discovered.

**Plans**

Going into these last few weeks I need to start worrying about the last few assignments for this term. This is primarily the end of term write up, the beta functionality for the code, and the demo video of our project. For this I plan to meet with my group members and then discuss how we want to start approaching these upcoming assignments. I want to get started on the auto connect on start up for the controllers and finish it this weekend. Like I mentioned I am meeting with the ECE team this weekend for two reasons. To confirm my integration of the code is correct and in order to get their code for the electronic Bluetooth part. I want to ask the ECE team this weekend when they think the finished controller will be available for me to test with. As having that controller will help with my integration part of the project. I also need to ask the ECE team about what exactly they want sent to the controller client from the raspberry pi server code. I want to focus more on general testing to try and catch any ideas I may have missed. For this I will ask my teammates who are creating the game to try and play their game using my client code next time we meet as a group. This is how I found out multi-button presses were not working as intended. I also need to begin working on the shell code that will try and keep the controller always connected to the raspberry pi while it is currently running. I will know if the auto connect on startup works this weekend, and if not, I can hopefully have the shell code to keep the controller always connected to the sever done by the end of the week. All of which I hope to report to client during the week 10 client meeting.

**Blog Post**

- March 13th, 2020
- Maxwell Evdemon
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

**Progress**

I learned this week that auto connect on start up will not be an issue, this is because I had to change all of my code this week. This week when the controller was handed off to me, I learned that the way the ESP32 runs, or at least how it was programmed to run is like a sever. This means I would have to re-write a majority of my code for it to be compatible with the ESP32. Note I referred to the ESP32 as the electronic part in previous weeks, as I did not know the proper name for it. So, with that in mind I made quite a bit of progress on integration. I was able to re-write essentially everything in a different format so it would essentially function as a client to the ESP32’s sever code. This took a little bit of work to do but it was not too bad as the general idea was typically the same. In terms of getting the controllers to connect on start-up, I was given a button directly on the console which I would be allowed to use. For this I have a new plan
to implement the code. As now my code is essentially a client code, this means I get to be the one who connects to the server, rather than trying to use Bluetooth modules to force a client to connect to my server. This makes my job quite a bit easier, as all I would have to do is request a connection to the ESP32’s server when the button is pressed on the console. This is something that ECE team is working with me to implement as it deals with pin’s directly on the console, which is something that is part of the ECE teams’ scope of work. Going back to the re-written code, I was able to have both controllers connected to my client code. However, I did run into some issue that I will note in the problems section down below. As a team we have begun to think about the upcoming video and end of term reporting. We decide to each record our segments of the project then edit them together. I am currently in the process of writing my slides and recording my video for beta testing. This week there was also a client meeting and an optional TA meeting. During the TA meeting we would explain to the client what we have done and what we were planning to do going forward. For the TA meeting I asked about the video and possible ideas I could use for my part.

Problems

My main problem this week was that I had to essentially rewrite all of the code I had been working on the entire term. This was primarily with the python code I had been using. After awhile of running the code would seemingly stop receiving messages from the ESP32. This was due to the ESP32 failing to send messages, as for some reason when connected to my python code the ESP32 would eventually just stop sending code string. This is something that could be confirmed from the ESP32’s terminal as well. Originally, I worked with the ECE member to get to bottom of the issue as we both thought it was a potential issue with the ESP32. Today however, we have a new idea to go off of, as it was found the python can struggle sometimes with the Bluetooth library and code. This is something that python3 does not have any issues with at all. I am not sure of the exact reason on why this is, but my plans are to implement this code going forward. There was another issue this week with integration as well, it came with the fact that sometimes I would receive the hex value for null in the code string. However, it is theorized that this issue is also something to do with python3 versus python. As there was testing done to for these two issues, and neither seemed to occur when running on python3, but this has yet to be fully proved.

Plans

Going forward I need to finish my part of the end of term write up and finish my part of the video as my primary concerns. I would like to work on the issue between python3 and python but completing this term’s assignments come first. After that I would like to do more testing with python3 idea to see if I can actually get it to bug out. As it may fix the ESP32 freeze issue, there is no guarantee that it also solves the issues with the hex-based null that keeps messing up my calculations. The hex-based null is an issue because of the decimal based functions I am using, and when it appears it causes the code to crash and I have been unsuccessful in solving it so far. When it comes to working on getting the controller to connect on startup, thanks to the button on the controller this is no longer and issue. As I can simply press that button and have it run my code to connect the controller. Something I would like to implement involves the LED light on the controller themselves. In case the controller disconnects for some reason I would like to have this reflected on the LED’s themselves. I think this would be nice visual way to understand If our controller is not working properly, this way we can run the sync code to connect the controllers once again. Speaking of the sync button on the console, at this point in time I am not entirely sure if it is working as intended. This is something I will need to confirm with the ECE team as they said they would look into the issue for me so I could focus on testing. I need to get multiplayer
working again, unfortunately the jump to being a client from a sever means I will not be using threads. So, I will have
to rework how I am currently doing multiplayer. I was able to confirm that it is still working when connected to two
different ESP32’s but I still need to add in the customization for each controller. I also have to run several tests again, as
the switch from sever to client has me worried that something may go wrong if I don’t test everything again.

11 WEEKLY BLOG POSTS – CORY HAYES

Blog Post

- October 18th, 2019
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Tristan and Max met with the ECE team today and were apprised of changes to the DreamZBox’s controller. The
controller will now have an accelerometer, a control stick, and 6 buttons (including pressing down on the control stick).
The accelerometer must be featured in some way in the game.

We met as a group this week and discussed the type of game we’re going to develop. We’ve decided that a 2D
adventure game with a top-down perspective would be fun. The game will also feature some procedural generation
in its level design. Our Problem Statement document needs to be finalized this week and we’ll begin writing the
Requirements document.

Blog Post

- October 25th, 2019
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

We clarified some points with our client and the ECE team this week. The controller used for the console has 4
context buttons, a ‘start’ button (usually for accessing game settings), a control stick for directional movement, and
an accelerometer. The console needs to be able to play games from USB and the interface we’re designing; this means
being able to install and store games on the console is a functional requirement. The console uses HDMI to connect to a
graphics display but doesn’t necessarily need to be compatible with a projector.

Next week, we will be researching the libraries commonly used to code games for the Raspbian OS in preparation
for our design document.

Blog Post

- November 8th, 2019
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz
Progress
We finalized our tech reviews this week detailing 12 major design aspects of our project, including: graphics assets, GUI frameworks, programming languages, game engines, music generation, image loading, sound effects, design philosophy, parallel processing, and more. No new changes have been made to the hardware.

Problems
Our Latex formatting for our documents hasn’t been consistent, so our TA suggested a format we should adhere to for future writings.

Plans
We need to decide on a development platform (at least for the UI) and a language to code in. The earlier we get something working on the Rasberry PI, the quicker we can assess the hardware restrictions imposed on our game.

Blog Post
- January 10th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

Each team member has completed critiques for the others regarding individual performance during Fall term. I then wrote an introspective review detailing how I wish to improve for Winter term. I am intending to make myself more available to communicate about the project and start assignments earlier. We met with our client and the ECE team this week. Megan clarified for me that we need to do our own poster design as the CS team. She also explained that if our game doesn’t meet all specifications, or doesn’t work properly by expo, that isn’t a complete failure. All the documentation and work leading up to the expo has its own merit. We also talked about the importance of developing talking points with prospective employers when they come to talk about our project.

I plan on reading some Qt framework documentation to learn how we can get some simple graphics to display this weekend. I’ll probably continue to work on Qt through the week. We’re going to have a brainstorming session next Tuesday (1/14) to discuss more game features. It’s a challenge learning everyone’s new schedule for Winter term, including our TA’s, but we should have our meeting schedule solidified next week.

Blog Post
- January 17th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

We had a brainstorming session this week to discuss what we are individually working on as well as any new features for the game. The console supports multiplayer, so we considered how two players would play our game. One idea is to have one player control the main character and the other control the monsters. Previously, we had trouble deciding what game mechanic would be improved through the use of the controller’s scroll wheel. This scheme would allow us to use the wheel to switch between monsters.
Initially, I planned on using the RPI emulator to test code on a virtualized Rasberry Pi environment, which is the platform our console is based on. However, because of the 256 MB RAM limitation of RPI, I am also going to research other methods of testing that allow utilization of the full 1 GB of RAM the Rasberry Pi 3 has.

Blog Post
- February 7th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

In our client meeting this week, we were informed that RPG Maker could not be used for building our game because it is not open source. We will instead shift toward developing for a web browser. Haofeng found a Javascript library called CreateJS that can supply graphics and animation methods. Our next steps involve incorporating our graphics assets using the library, creating animations, and controlling them through keyboard commands. Our control scheme will need to be flexible enough to accommodate analog control sticks rather than merely cardinal directions.

Design reviews start February 18th, and we will be presenting a working game running on the Raspberry Pi (console). Max has developed the bluetooth connectivity to enable his phone to act as a controller for the simulation.

Blog Post
- February 14th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

I completed the first animated graphic for the game using the CreateJS Javascript library. Next, I’ll be working on creating a map for the character to move in for our demo. Reading map data from a file will not be possible as Javascript does not have direct access to the file system, and we don’t want players needing to select a map manually. So, I will need to create maps procedurally. I would like to have map generation and the ability to move between maps complete before the design review.

We are going to meet next week to discuss our presentation for the review. We will also be meeting with the client and ECE team for group pictures on Wednesday 2/19.

Blog Post
- February 21st, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

We met as a group to discuss our design review and complete our slide show presentation. Topics for our presentation include: design of dashboard and game, console controller implementation, and our research process. We also met with the client and ECE team to take pictures for our expo poster.

I will continue to work on map generation for the game in the coming week. I also need to develop interactions between the player character and enemies.
Blog Post

- February 28th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

We had our design review on Thursday where we explained our project to 4 other capstone teams. My part of the presentation included describing the knight animation and map tiles used in the game. I’m continuing to work on the map generation for the game. There are bugs causing some of the map tiles to be undefined. After I get that worked out, I’ll move on to navigation between maps.

Blog Post

- March 6th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

I finished up my procedural map generation code. There are some bugs to work out, but I feel that pseudo-random maps will make the game more dynamic. Tristan implemented some enemy AI and Haofeng wrote some collision rules. I’m confident we will have some game play by the end of next week. Next, we need to work on our demo video that is due at the end of next week. I will be doing a Kaltura capture of some of the game play while I explain graphics and map generation.

Blog Post

- March 13th, 2020
- Cory Hayes
- Project: DreamZBox 2.0
- Team: Dreamz Catcherz

We are completing our demo video and report for the end of the term. I discussed the algorithms for creating maps for the game as well as animations. I still need to find a solution for the player traversing room to room. Currently, one room isn’t linked to any other. Even though our game is not feature complete, we are working towards having the characters interact with enemies and complete the maze.
13 PROJECT DOCUMENTATION

13.1 Maze of Dreamz

13.1.1 Running The Software

The links to the game’s repository and guide are in Appendix 1. The does not need to be compiled, installed, or built. It is a simple HTML, CSS, and JavaScript application. It needs an HTML5 compliant browser environment to run. Opening the “index.html” file with a browser will run the game. The “index.html” file attaches the game code within the “head” element of the HTML document. These can be seen with the series of script tags. The order of the files are significant as well. These files are loaded from bottom to top. One could navigate through the project this way to trace through the game. This is the best way to understand the game.

13.1.2 Architecture Description

The game uses a Entity Component System architecture. This is described in Tristan’s Tech Review. The game combines the concepts of components and system. Each system has a registry. This registry represents indexes/identifiers to lookup the entities/sprites in the game to operate upon them. The game runs by rolling through each system, looking through their registries, and operating upon every sprite in the game. This architecture could use additional correction for consistency across systems. The game does allow for asynchronous and parallel operation with this format. The file “src/systems/system.js” defines the pattern. It allows for functions like that in “src/systems/system-template.js.” The system paradigm is important to the game. It consists of most of the programming.

The game engine does slightly cheat this design. The games generation of the map was made before the architecture was set in stone (and designed). The entities would have to be statically created in some format, but the behavioral composition of entities would not typically happen in this architecture. This would be changed in future iterations of the game.

13.1.3 Gameplay

The game uses standard computer controls with the WASDE keys. These allow the player to move around and attack with the knight. Any user can also refresh the page to see the procedural map generation in action!

13.2 DreamZDash

13.2.1 Installation

The DreamZDash is a C++ program and therefore requires compilation. It has been programmed for use on a Debian/Raspbian Linux system. This operating system is common for Raspberry Pi machines. This makes it very hard to test on non-raspberry pi systems. It also requires graphics so it cannot be run over an SSH terminal session. The directions for installing the game are in the repository found by the link in Appendix 1.

The instructions mainly act to install the prequisite software: GCC, SFML, and CMake. Feel free to install these on your own and use the “build.sh” file to build the project. This will automatically copy over the assets (pictures) for the dashboard. All of these processes may be accomplished manually.
13.2.2 Using The Dashboard

The dashboard uses the same WASD keys for traversal like the game. It then uses the enter key to select an option. However, the ability to select widgets on the screen is removed due to bugs at the time of writing this. The dashboard currently uses placeholder information for what would be the games. The vision for this was to have metadata available for users. The metadata would be stored in a file within a hard-coded directory and used as a visual. The transversal of widgets on a screen with a blue square encapsulate the entire functionality of the dashboard.

13.2.3 Architecture

The architecture within the design document (section 3.12.2) matches the program architecture. Each widget was broken down into an object and given a set of properties to abstract it’s own behavior. Each widget would then be loaded with a lambda callback function to perform some operation upon the dashboard. While the constructors leave room for these callback functions, the dashboard does not have these defined. Other processes from the design document were attempted but never finished so instead an object oriented dashboard front-end was the only thing completed at the time of writing this.

13.3 Bluetooth

13.3.1 Installation

The connection itself is mean to run on hardware, in this case that hardware would be a raspberry pi with the proper libraries installed and with Bluetooth connectivity. Installation for the Bluetooth connection is primarily installing all of the necessary libraries needed to run the code. The code runs using python3 with the keyboard and Bluetooth libraries. The keyboard library will need to be installed to the root to work properly. The system filedbus-org.bluez.service for raspberry pi Bluetooth will also need to be modified in order for the connection to work. An in depth installation guide for the Bluetooth connection can be found in Appendix 1 in the MazeOfDreamZ GitHub in the Bluetooth connection directory.

13.3.2 Using the Bluetooth Connection

Running the code will require one of the created controller from the ECE team or a configured ESP32 that sends serial messages. Make sure the raspberry pi that is being used has it’s Bluetooth powered on and is register to the controllers through the bluetoothctl module. If the controller is not registered, one can access the bluetoothctl module in the raspberry pi terminal. To start the connection first power on the controller then run the client code on the raspberry pi terminal. Doing this should cause a light to appear on the controller that is powered on, and the terminal where the python code is running should begin to scroll due to taking inputs from the controller. Moving the joystick on the controller or pushing buttons will then cause the keyboard inputs to appear on the tab you are currently on. If using the original raspberry pi that was provided by the client, next to the power button there should be a sync button. Pressing this button will cause the client python code to be run, as if they are run in the raspberry pi terminal.

13.3.3 Bluetooth Connection Design

The python3 code acts as a client to the ESP32 server inside the created controller. The python3 code will try to connect to the controller when ran by send a request and will timeout after a few seconds if it cannot reach the server. This is why the controller must be powered on before the running the python code, so the client can reach it’s destination. The
serial string received by the client code from the server contains the necessary information to determine what keyboard inputs are interpreted. The keys on the virtual keyboard are pressed depending on this serial string. The key are held down for a short amount of time, which allows for the application that receives the keyboard input to register which keys are being pressed and what action it should take. This press time is kept short so inputs from the controller will not be ignored. Then the program will release the key that are pressed, which is can determine using the original serial input string. Doing this allows for multiple key to be determined at one time, allow the player to commit to multiple actions.

14 Recommended Technical Resources for Learning More

The bibliography at the end has some great references used during design and technology review for reading. These articles pose great introductions to technologies within the creation of DreamZBox 2.0.

14.1 Maze of Dreamz

The game uses CreateJS and Vanilla JavaScript to create the entirety of the game. These tools also have dependencies like WebGL which bookworms may also research into. The following links represent databases and websites with reference material and introductions to these concepts.

- W3Schools Javascript Tutorial: https://www.w3schools.com/js/default.asp
- ECS Talk: https://www.youtube.com/watch?v=W3aieHjyNvw

14.2 DreamZDash

The dashboard uses SFML, C++, CMake, and Object Oriented design. These are fairly easy and common to read up on. They are also pretty broad concepts which makes understanding them pretty simple. The following links explain the concepts fairly well.

- SFML Introduction: https://www.sfml-dev.org/tutorials/2.5/
- CMake Documentation: https://cmake.org/cmake/help/v3.17/
- Object Oriented Design: https://en.wikipedia.org/wiki/Object-oriented_design

14.3 Bluetooth

The Bluetooth coding requires knowledge of how Bluetooth with Raspberry Pi works. It also requires knowledge of Python Standard libraries to initiate communications over Bluetooth networks. The following documentation links provide background knowledge on these subjects.

- Raspberry Pi Bluetooth Introduction: https://www.electronicwings.com/raspberry-pi/using-raspberry-pi-3-
- Socket Networking Docs: https://docs.python.org/3/library/socket.html
- Python Tutorial: https://www.tutorialspoint.com/python/index.htm
15  CONCLUSIONS AND REFLECTIONS

15.1  Haofeng’s Reflection

15.1.1  What technical information did you learn?
I have learned about CreateJS and learned how to use JavaScript library to build simple games, the logic of collision function in game.

15.1.2  What non-technical information did you learn?
Through the showcase mini assignment, I have learned how to deliver the project to different groups of audience. The group presentation and code review are good experience for me to learn how to communicate with people in different technique fields.

15.1.3  What have you learned about project work?
We were making the project based on clients’ requirement. To make a good project we need to keep communicate with our client, know what they want and whether they are satisfied with our choice.

15.1.4  What have you learned about project management?
We need to always to plan ahead of time, make the time schedule before starting the project, and try to leave some spare time as possible for some emergency cases. In this way we can have our project be well organized.

15.1.5  What have you learned about working in teams?
It is important to keep exchange the idea with teammate, when we want to make any changes or want to add something new to the project, we also need to let the team member know in the first time. Thus we can avoid the repeat work or the conflicts.

15.1.6  If you could do it all over, what would you do differently?
The project requires huge amount of time to complete, if I could do it all over, in the first term I would spend more time on analyzing designing and coding stuff instead of only focus on the documentation. And I will communicate with the clients more often to make sure I’m on the right track, to avoid the rejection of design choice.

15.2  Maxwell’s reflection

15.2.1  What technical information did you learn?
For technical information I learned more about the inner workings of the raspberry pi, specifically on how it handles it’s Bluetooth libraries and connections and how it decides which programs to initialize on start up. From the technical reviews I was able to learn more about USART, I2C and HDMI functionality. I also learned about the python Bluetooth libraries and how the connections are typically set up as client and server. This includes libraries such as pybluez which are used for Bluetooth control. For the server based code I learned more about threading how it can be used with Bluetooth. I learned from hardware such as the ESP32 I learned more about arduino code, programming, and the hardware.
15.2.2 What non-technical information did you learn?

I learned more about the presentation and writing of technical information and documents. Particularly when talking to certain demographics or people who may not know as much about the projects as oneself. I also got more experience in adapting to situations when necessary and the need of being flexible with under those situations. Always properly research a task you are working on and make sure you have alternatives before starting that task.

15.2.3 What have you learned about project work?

I learned that project work should not be taken lightly, especially when one does not have time to divert there full attention to that project. If one is working with hardware always have some sort of backup hardware just in case the hardware is faulty. Also plan ahead more when working with hardware that only will function given certain conditions, or have a reliance on other pieces of hardware.

15.2.4 What have you learned about project management?

For project management, I learned that one should always plan ahead for when working on a project. I learned that you should always have sort of backup plan just in case the original idea is impossible. When working with a code based project try to follow some software development techniques like scrum or waterfall. Always have members outside your group test your project, they can often find errors the developers did not think of during the creation phase.

15.2.5 What have you learned about working in teams?

Communication is key for any team based project, this becomes especially clear when working in larger groups, which multiple sections like the ECE team and the CS team. Keeping an open stream of connection with group members and the clients is the only way to make sure there is no duplication problems or wrong information. I learn that one should always keep a back up of their project in a way that allows teammates to have access to their code. This way they understand what they are working on, and have access to the code for questions and testing.

15.2.6 If you could do it all over, what would you do differently?

The key thing I would do better is communicate better. I had several issue primarily due to poor communication on my part, with the most infamous being that I had to refactor my code during week 10 of winter term. This was due to thinking the ECE team was building client based code when they actually built sever based code. Open communication could have avoided that situation entirely. Second always have some sort of schedule to try and keep up with. I feel like I could have finished the project more efficiently if I held myself to set dates when certain parts of the project that to be done. Next would be to always look for ways to improve on my part even when waiting on the output from another teammate. Specifically in this case, I was waiting for the prototype controller to be done for testing, when I probably could have found some of the bugs with what I already had. Fourth, I should have set aside more time for integration testing with the ECE team. While it was completed, it didn’t need to be finished so close to the due date beta functionally during winter term. Last, I would test more on the given hardware from the clients. In this case, I did most of my testing on my personal raspberry pi and once that was complete I assumed it would work the same on the client’s raspberry pi. This was not the case, which results in more time spent debugging and testing.
15.3 Cory’s reflection

15.3.1 What technical information did you learn?
I learned about the CreateJS Javascript library and how it can be used to make sprite animations. I also learned how to work with the HTML5 canvas. I hadn’t worked with 2D animations before this class.

15.3.2 What non-technical information did you learn?
I learned a bit about explaining software to a knowledgeable audience during our Design Review. We don’t get many opportunities through our college classes to explain our work to peers in a professional setting. Prior to going back to college, I was usually required to explain software in very general terms to users and supervisors. The risk of overusing technical jargon is much less when talking with peers.

15.3.3 What have you learned about project work?
This is the first project I have done that required so much documentation up front. I learned that it is imperative to remember what is promised for the project, and to recognize when your roadmap to completion changes. At this point, the client should be updated as well as documentation (even if it is not required by an assignment).

15.3.4 What have you learned about project management?
I learned that sometimes it is better to cut your losses when researching a new technology if it appears it is not going to work. Before I learned about the graphics limitations of the Qt framework, I had other signs that it might not be suitable for our project. Qt had an editor that was very large in size and many of its libraries were the same way. The longer I spend trying to get something to work, the harder it is to let it go.

15.3.5 What have you learned about working in teams?
I learned it is much more important to start work early when working with people with disparate schedules. I procrastinate too much, and I don’t always get my ideas on paper quickly. In teams it is much more important for everyone to see what you’re thinking so the discussion can begin.

15.3.6 If you could do it all over, what would you do differently?
I should have realized how much time this project was going to take and limit other commitments. Tell the client about every minute change in the roadmap, because I might be unaware of the significance of a detail. We started working with a free, but not open-source, graphics engine at the beginning. This needed to be scrapped because I didn’t realize it wasn’t open-source.

15.4 Tristan’s Reflection

15.4.1 What technical information did you learn?
I learned about SFML and CMake when working on the DreamZDash. With Maze of Dreamz, I learned how to apply ECS correctly. I had yet to do that in a professional capacity, so that was insanely fun. Trying to apply VanillaJS to a very in-depth system is fun. Additionally, I got to see an advanced graphic package in action. To put this altogether gave me confidence to create a Hackathon project called Shahnoza. This was a great experience. I also now realize the importance in using as much libraries as possible rather than rewriting the wheel. Node Package Manager (NPM) actually has great tools for Browser Based Applications. All of these technologies were available for me to learn because I could embrace them on my own.
15.4.2 What non-technical information did you learn?

I learned how to design complex architectures. Reading the documents on design languages and parameters helped me further understand how to create a product from scratch. This has helped me work with others on side-projects. I am actually working with a high schooler to create a small Python application. With the ability to express in writing the design of an application, the student and I work very well together. I wish I learned more methods of working with other developers on a project. I feel like I either over complicated or oversimplified pieces of the project without knowing how to compass the project. Knowing how to boundary set correctly, I would have had better benchmarks for the project.

15.4.3 What have you learned about project work?

I have learned that projects are marathons not sprints. A project should not be sought after in a few sprints. It should be expanded with time for people to breathe and rest properly. I also learned that enforcing concepts for a project should be done early on so that the development does not slow later on. Being shady with the clients will only end in false positives and project failures. Working with the clients is an epic skill and I will continue to work on it.

15.4.4 What have you learned about project management?

I have learned that project management does not become realized with a SCRUM board alongside a few hopes and dreams. Project management is about working with your cohorts to navigate the project in their way. It does not matter what can be done. It matters how you can use what the programmers’ skills to create a product. Just because a project is possible does not mean it is plausible.

15.4.5 What have you learned about working in teams?

I have learned that working in a team is only fun if everyone believes in memes becoming dreams. Otherwise, the team may suffer from low morale. It’s also important to adopt other developers’ styles and operations. One developer on the team taught me about the importance on Pull Request comments. This is a new skill I learned from working in a team. I can now apply this to other teams.

15.4.6 If you could do it all over, what would you do differently?

I would have just done a game with Phaser IO. I think that creating a Game Engine is too large scale for a project. It would have also been simple to have a project that can be recreated in web environments like REPL IO. I also think that I would have chosen to do Capstone at a time where I had a lower class load. I found that I enjoyed spending the last year of my college experience doing things outside of capstone which decreased my free time. It would have been better to taken an additional year of school rather than graduate early.
REFERENCES


16 APPENDIX 1

Below are the GitHub Links for the source code of the project.

- **Maze of Dreamz**: https://github.com/TFlexSoom/MazeOfDreamz
- **DreamZDash**: https://github.com/TFlexSoom/DreamZDash

Below is a playable version of the game online hosted by the ONID servers. The link will most likely go bad on September 2020.

- **Maze of Dreamz**: http://web.engr.oregonstate.edu/~hilbert/MazeOfDreamz/
The following were the original Pencil Paper Designs created by Tristan Hilbert.
Debug Mode ⇒ Show Input Tags

Comments:

Notes:
Title Screen

Comments:
We need to decide between Cnt renders and image renders.

Notes:

Entities

Comments:

Notes:

Maze of Dreams

Particle effect

play single

play double

select

Effect player?

gold

gold

Flat

Wall

Path

Entity

Point/Shoot

Freeze

Block

Boss
Attack / Spell / Animations

Comments:
Test which abilities are fun.

Notes:
See key effects once.

Example of Play
Comments:

Notes:
Camera focused on first player (first player preference)
Selector in upper left

Element Selection Panel
Comments:

Notes:
Taken out!!
No longer intended

NOT ACTUAL IN GAME IMAGE!!
Camera is adjusted!
Comments:
- Particle effects will require custom graphics.
- Tornado will move, destroy after 3 seconds.
- Bobbles stay for 0.3 sec.

Notes:
- * Rules with traps:
  - Only 4 at a time.
  - Explode on Energy Collision.
  - Explode diagonally for 0.2 sec.
  - Delete after explosion/respawn.
- Bobbles stay for 0.3 sec.
- Slow/hill enemies.

PAUSE
(Continue)
Quit

Comments:
- Notes:
- Add two lines and circle for upper right for player 2.

Comments:
- Should we include a "drop play-2" option?
- Notes:
18  APPENDIX 3

The following pages came from the Code Review held for the group.
<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Reviewers Comment</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Build</strong></td>
<td>Could you clone from Git and build using the README file?</td>
<td><strong>Group 27</strong>&lt;br&gt;Yes, it is easy to clone the project from Github. It needs some commands to install and it is clear to learn how to build that you just need to follow the instructions.&lt;br&gt;&lt;br&gt;Yes. I could clone the project folder to my own machine and make files using the instructions.</td>
<td><strong>DreamZDash</strong>&lt;br&gt;- Add capabilities to DreamZDash for building on Ubuntu&lt;br&gt;- Correct Spelling and Grammar Errors in instructions &lt;br&gt;&lt;br&gt;&lt;strong&gt;MazeOfDreamz**&lt;br&gt;- Add control information to the game itself&lt;br&gt;- Have pre-built version for testing purpose</td>
</tr>
<tr>
<td>Group 27</td>
<td>Group 28</td>
<td>DreamZDash</td>
<td>MazeOfDreams</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Yes, and the code adheres to general guidelines and code style.</td>
<td>Yes. Adding more comments will be a good idea.</td>
<td>- Add function headers</td>
<td>- Adjust some naming schemes to provide some consistency</td>
</tr>
<tr>
<td>All the files are easy to follow. Code style looks good and makes the files clean.</td>
<td></td>
<td>- Consider a style guide for javascript</td>
<td>- Comment any new functions not made obvious through name</td>
</tr>
<tr>
<td>There were differences in naming conventions, like in render.js some variable names started with render and others ended with render, it’s a small thing that can make a huge difference for reading it :)</td>
<td>There doesn’t seem to be a code guideline Having a code guideline would be a benefit to your project</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Able to access the MazOfDreamz/index.html webpage to play the game
- README had instructions to play games

Legibility

Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?
<table>
<thead>
<tr>
<th>Implementation</th>
<th>Group 27</th>
<th>DreamZDash</th>
<th>MazeOfDreamz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes, it is shorter/easier/faster/cleaner/safer to write functionally equivalent code. There is no dead code and the comments clearly explain the functions.</td>
<td>- Continue to abstract carefully &lt;br&gt; - “Polish” more</td>
<td>- Create a glossary for things like “reg_object” to keep code concise without using verbatim name.</td>
</tr>
<tr>
<td></td>
<td>There are no super long and complicated methods in the code so I believe the team has been doing great on keeping the implementation easy to follow.</td>
<td>The comments do not express any implementation changes. This is a huge bummer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Group 28</td>
<td>Overall it is good, I think it could be a little more polished, but it is good!</td>
<td></td>
</tr>
<tr>
<td></td>
<td>So far so good.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Functions look organized and code looks clean. Functions are spaced</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Headers and comments on most file &lt;br&gt; - Some have just headers &lt;br&gt; - Some functions I don’t recognize don’t have comments</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Most variable names work, but some are unrecognizable with initial look
- I see `reg_object` a lot in the systems files. Not sure if its `regular_object` or maybe `register_object`? Not sure what it exactly does

<table>
<thead>
<tr>
<th>Maintainability</th>
<th>Group 27</th>
<th>DreamZDash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there unit tests?</td>
<td>Yes, they have the unit tests and it also needs to have the tests. Due to this being a game, the unit tests can help to learn the bug and fix it.</td>
<td>- Add Unit Tests</td>
</tr>
<tr>
<td>Should there be? Are the test covering interesting cases? Are they readable?</td>
<td>I didn’t see a test file in the github repo but I believe they are going to have it done in the future. Unit tests are important for games since there could be many edge cases.</td>
<td>- Pick a Testing Framework</td>
</tr>
<tr>
<td></td>
<td><strong>Group 28</strong></td>
<td>- Consider edge cases</td>
</tr>
<tr>
<td></td>
<td>I don’t see any unit tests. I think there should be unit tests, expecifcly for a game.</td>
<td>- Use the sections mentioned for potential ways of testing</td>
</tr>
<tr>
<td></td>
<td>I couldn’t find any unit tests, and I think that the project would benefit from that</td>
<td>I am disappointed none of the groups suggested frameworks or cases to test in specific. I admit this is not required, but I do not see other suggested actions from “add unit tests”.</td>
</tr>
</tbody>
</table>
There is a section talking about potential ways of testing which was nice, gave a picture of the expectations of the program

- Appears cases are either covered by if statements or using switch cases.
- I see a lot of these cases, but that makes sense since there are game systems in place

### Other

Are there other things that stand out that can be improved?

#### Group 27

It is a really interesting game and I am excited to play it. The spritesheet is so cute and the “AttackEffect” is cool. I think maybe let the game be full of the screen which makes the UI look better.

I noticed there are many empty files in the repository. Not sure if those are still going to be implemented. It would be nice if you remove unnecessary files so that it won’t confuse the reviewers.

#### Group 28

Adding an “overview” to the readme and maybe having an outline/ of the code

Seems to achieve requirements. MazOfDreamz can launch the game. Was having trouble trying to run DreamZDash on Ubuntu.

### DreamZDash

- Create Full Screen Effect
- Add outline to README

### MazeOfDreamz

- Add Fullscreen setting to game
- Add outline to README