Overview

This document details the top level objectives, implementation, and interface details of the finale in order to facilitate effective communication among the Escape Room development team.

Requirements

Room

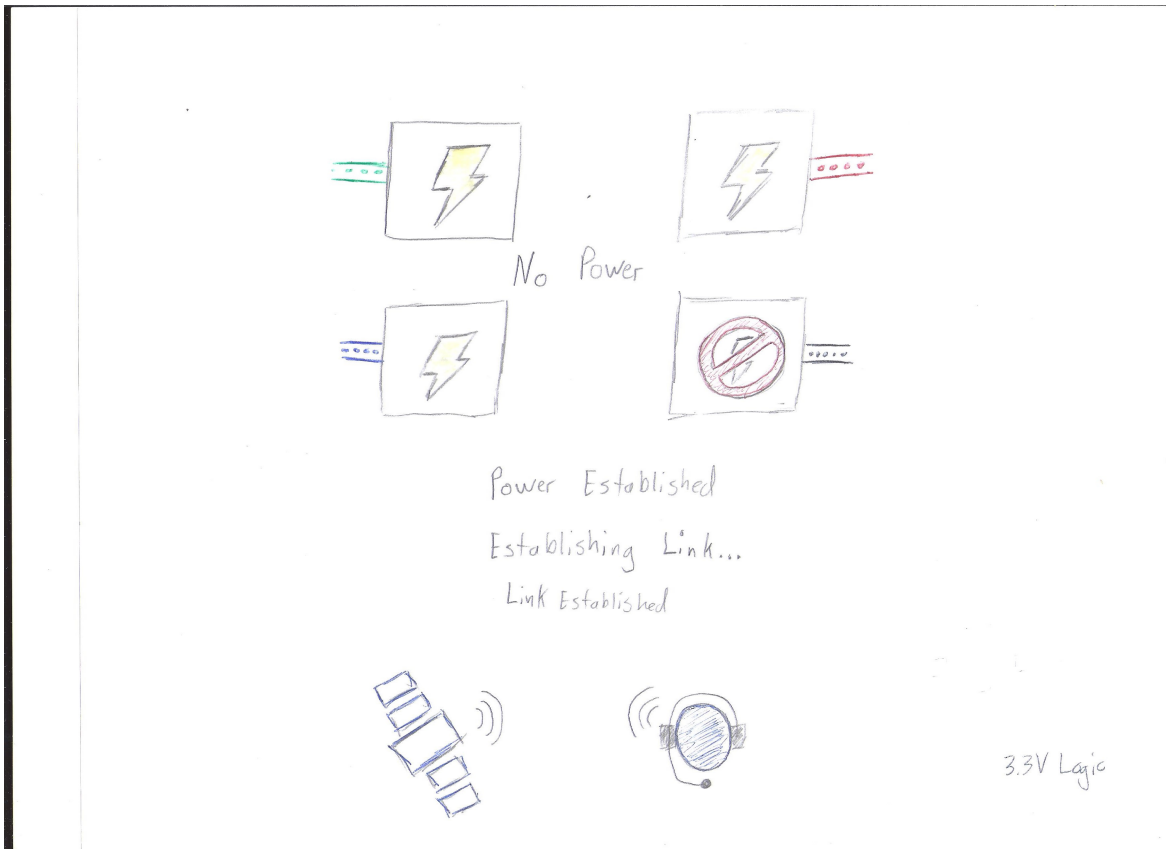
- Requirement: The finale shall follow a pre-scripted narrative.
    - Implementation: The wall computer will play video with sound as dictated by the narrative.
- Requirement: The goal of the finale shall be an immersive audio-visual experience that impresses players.
    - Implementation: Escape rooms are typically designed around the setting of graphic adventure puzzle video games common in the 1980's and into the 1990's. Escape rooms often leverage several inter-dependent physical puzzles that take one or two players to solve. However, using arcade and platform video games, such as Snake and Crash Bandicoot, as inspiration for the puzzles leads to an experimental multi-person real-time activity that differs from mainstream escape rooms.
    - Implementation: Music will be played during the Snake game, Hyperspace game, and Credits sequences.
        - Requirement: Music shall not break the space-themed illusion of the room.
            - Implementation: Instrumental music will be originally composed that fits the space-theme. Music will be designed to serve as a background accompaniment to the main player action, as is done to set ambiance in Hollywood movies.
        - Requirement: Music shall not be be too loud.
            - Implementation: Allow the proctor to adjust volume levels prior to and during room play-through.
            - Implementation: The volume for music will be independent from the volume for sound effects.
        - Requirement: After the Light Puzzle is completed, no other puzzles shall emit sound during the finale.
- Requirement: The goal of the finale shall be to encourage team work among the players.
    - Implementation: The Snake game will be most easily beaten if the players coordinate so only one on-screen character completes the objective at a time (to minimize the risk of players running into one another) rather than all proceeding independently at the same time.
    - Implementation: The Hyperspace game controls act as an Ouija board where multiple players need to coordinate their actions to achieve the desired outcome. Additionally, players will be presented with asymmetric controls and information: a limited number of players will be able to see the map to determine how best to reach the end goal; some obstacles like asteroid rubble must be avoided (one set of controls), while other obstacles like large asteroids cannot be avoided and must be destroyed with the laser (joystick in another station).
-

Requirement: The cameras, proctor computer, wall computer, and console computer shall all be connected together in such a way that near-real-time communication (latency shall be no greater than 100 ms) can occur between them, for example through Ethernet communication.

- Requirement: Custom electrical connections such as with the joystick and buttons will use heat shrink to strain relief the custom connections and reduce the risk of inadvertent electrical contact.
- Requirement: The wall, console and proctor computers shall be supplied with 120 VAC, 60 Hz power except in extenuating circumstances. Normal play-through and room reset activities shall not disconnect power from the computers.
    - Implementation: The computers are architected to return to a standby state after a completed play-through and wait for proctor input.
- Requirement: The wall, console and proctor monitors shall be supplied with 120 VAC, 60 Hz power except in extenuating circumstances.
- Requirement: A manually-accessible power switch(es) shall be available for the Proctor, Wall and Console computers, but not readily apparent to players. This may take the form of a power strip that is only accessible from below a table or behind a wall. This provides an easily accessible failsafe recovery option for the proctor to use should a computer become non-responsive.
- Requirement: When power is applied, the wall, console and proctor computers shall boot to a standby state in no more than 5 minutes.
- Implementation: The Wall PC will act as MASTER to the SLAVE Console. The Proctor will act as MASTER to the SLAVE Wall.
    - Requirement: The Proctor shall not command the Console directly. Changing the Console state could lead to an undefined state existing between the Wall and Console computers.
- Requirement: Following the completion of the Light Puzzle, Snake game, Hyperspace game, Credits, or any cut scenes, the Wall computer shall automatically progress to the next chapter, updating the Console computer as dictated by the narrative.
- Requirement: The wall monitor shall display the remaining time to solve the room prior to the end of the 60 minute window. Following either a successful or failed attempt to play through the room, the total time taken shall be displayed on the wall monitor until the end of the credits.

Light Puzzle

- Requirement: Following a start command from the proctor, an introduction cut scene shall play.
- Implementation: Mock up:



- Requirement: The wall monitor shall display the remaining time to solve the room.
- Requirement: The light puzzle shall provide four discrete signals at 3.3V/0V, and the corresponding discrete ground returns (a total of 8 wires), to the wall computer to indicate the status of the puzzle components.
- Requirement: The wall monitor shall display the state of the four light puzzle components as each either solved or not.
- Requirement: The wall monitor shall display a list of top level objectives for players to complete in order to solve the room.
    - Implementation: This will take the form of three bullet points: establish power, stock cargo pod, deliver cargo pod.
- Requirement: Once all four light puzzle discreets have changed state, the finale shall be considered finished.
    - Implementation: The Wall computer will then automatically transition to the next chapter, such as a cut scene.
- Requirement: A monitor that accepts and displays visual input from a Raspberry Pi shall be installed on the wall.
- Requirement: The wall computer shall be configured to run at 1920x1080 60 Hz (16:9, DMT mode 82) resolution.
- Requirement: A cut scene shall play after the completion of the light puzzle explaining the controls and objective for the Snake and Hyperspace games.

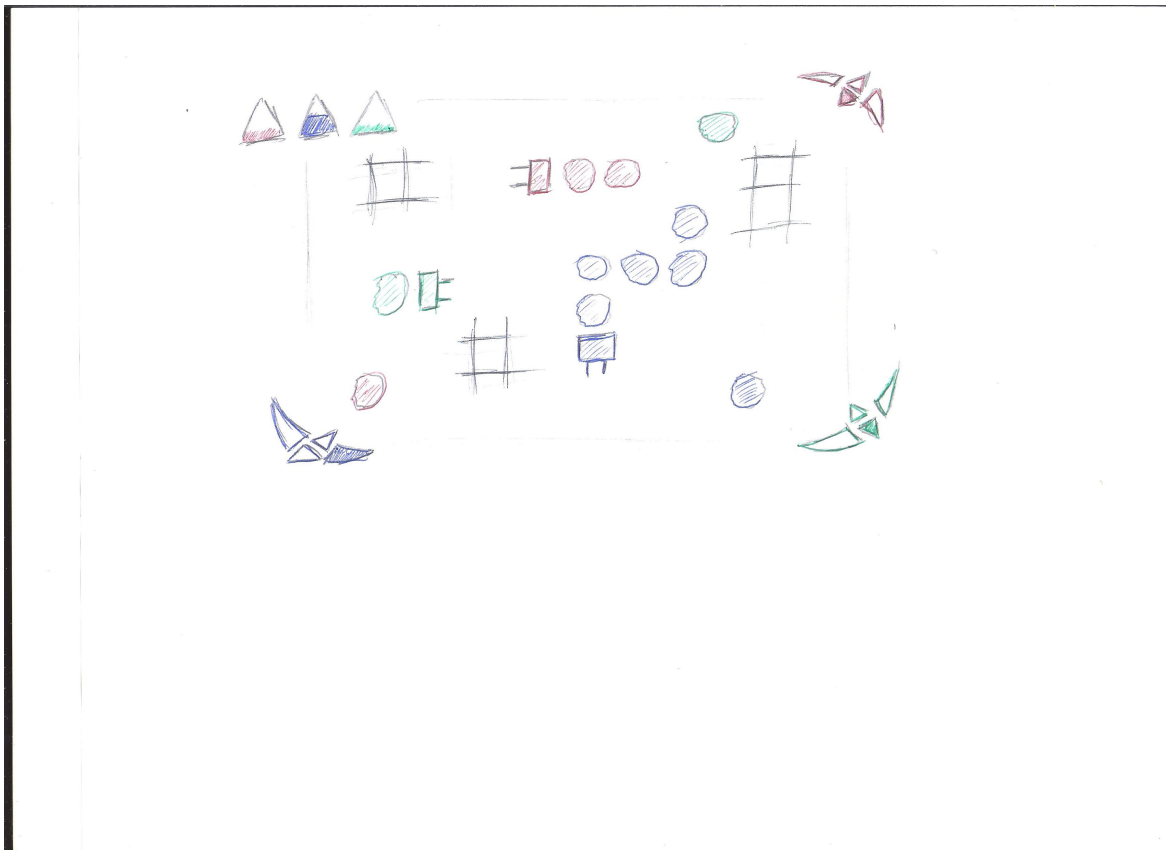
Morse Code

- Requirement: Once the light puzzle is solved, the wall computer shall send a trigger signal to the console computer. In response to this command, the console computer shall proceed to a blank screen within 1 second in preparation for the map portion of the Hyperspace game.
    - Implementation: The book state machine running in the console computer will cease updating the Morse code puzzle and rendering graphics in response to the relevant Ethernet command from the wall or proctor computers.

- Requirement: A monitor that accepts and displays visual input from a Raspberry Pi shall be installed in the console.
- Requirement: The monitor shall be situated in the console such that no more than two players can clearly see the screen.
- Requirement: Two buttons shall be installed in the console adjacent to the monitor to represent 'dot' and 'dash' user inputs. These buttons shall be electrically connected to the console computer so that the state of the buttons can be acquired.
- Requirement: The 'dot' and 'dash' buttons shall be situated such that a seated player can view the monitor while pressing the buttons.
- Requirement: The console computer shall be configured to run at 1920x1080 60 Hz (16:9, DMT mode 82) resolution.

Snake Game

- Implementation: Three snakes are shown on screen. One is controlled by the joystick in the captain's chair. One is controlled by the four buttons in the console. One is controlled by the four stomp pads. The corners of the screen provide visual user feedback of user inputs.



- Implementation: The music for this section will be inspired by the following reference songs:
  - Epic Mountain - Quantum Computers
  - Software Inc - Trailer
  - Traktion - Mission ASCII
- Requirement: The Snake game shall be designed to be completed within 5 minutes, but allow for shorter or longer play times depending on player skill level.
-

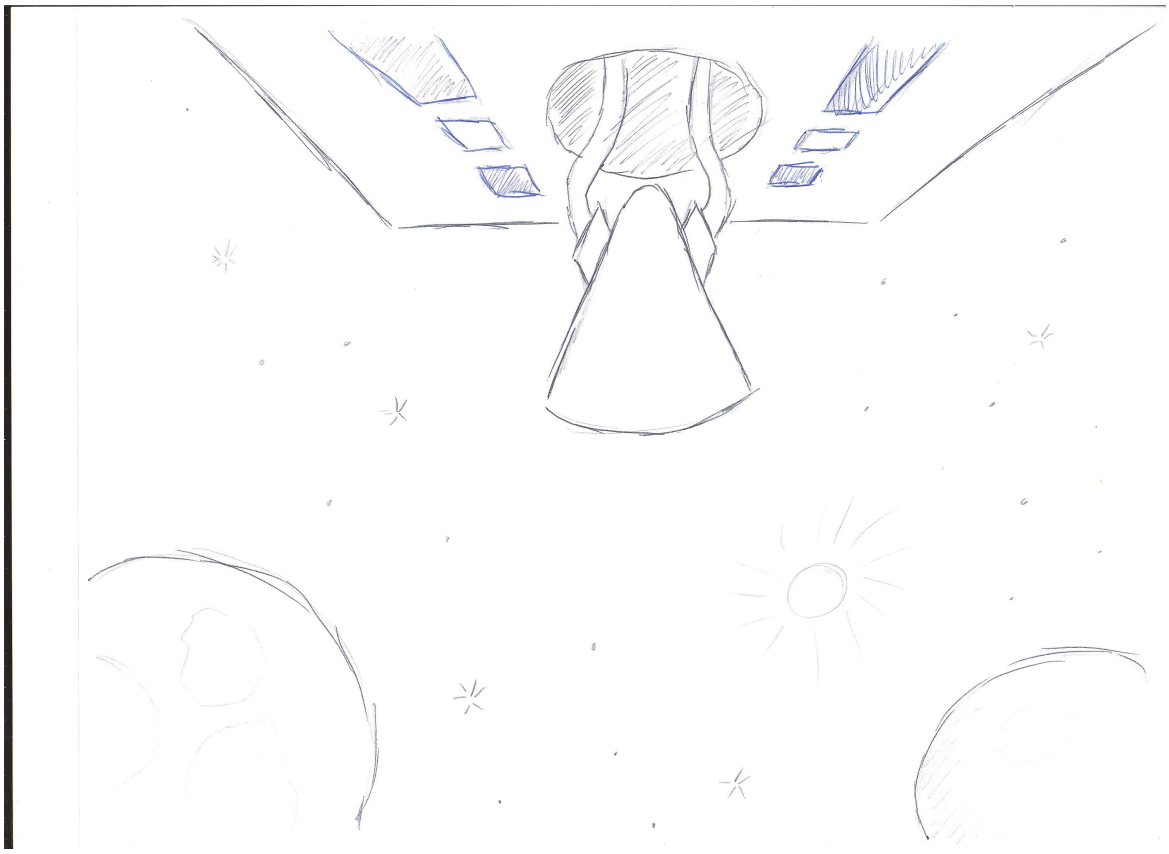
Requirement: The following user inputs shall be installed in the following locations and be electrically connected to the wall computer:

  - Four buttons in the console. These buttons will represent up, down, left, and right commands from one seated player. Each button has a sense and return. 8 wires total. These buttons serve as the 'blue' user input and the goal shall be to use this theme color when designing the mounting structure in the console.
  - A joystick that can be operated while a player is seated the captain's chair. Each direction, plus fire, has two wires for sense plus return. 10 wires total. This joystick serves as the 'red' user input and the goal shall be to use this theme color when designing the mounting structure in the captain's chair.
  - Four stomp pads arranged as two pairs of two. One pair of pads represents left and right commands. The other pair represents up and down control. The pads are designed to be operated by players while standing. Each pad contains three wires: power, sense and return. 12 wires total. These four pads represent the 'green' user input and the goal shall be to use this theme color when designing the stomp pads.

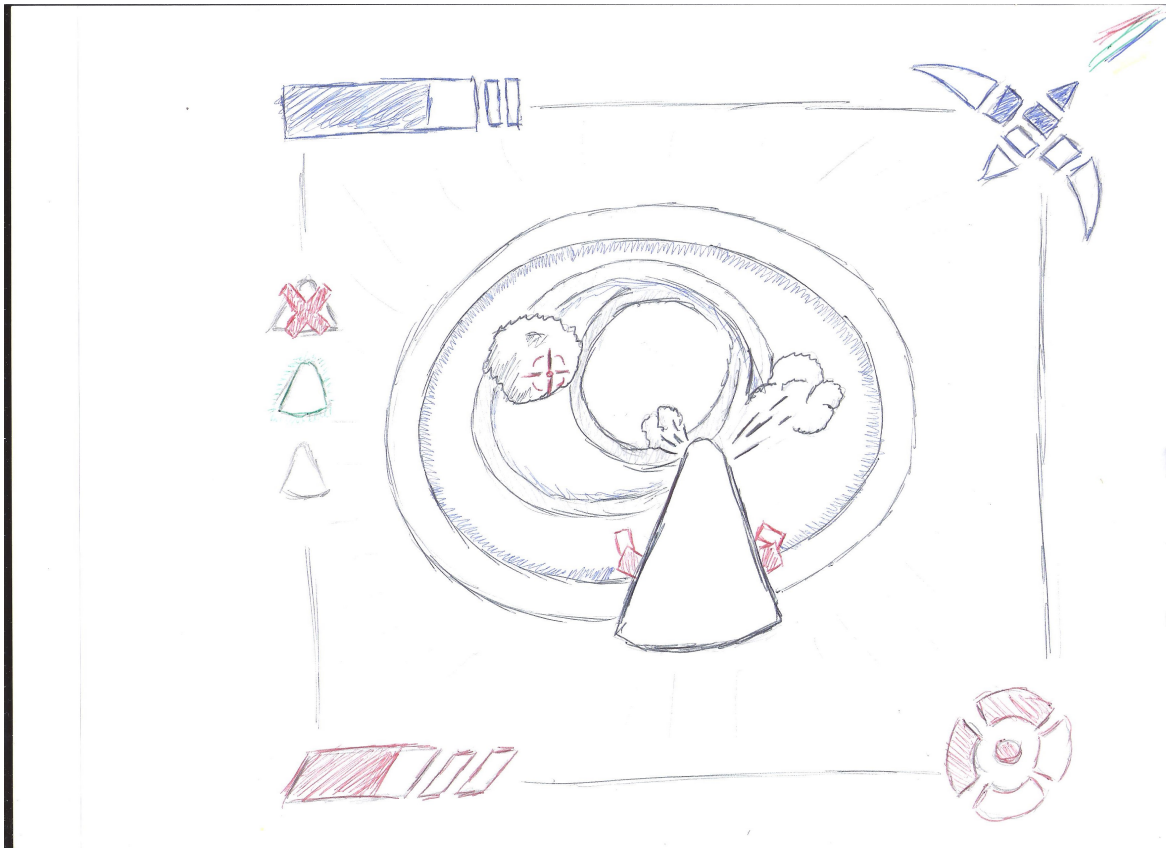
- Implementation: When one snake hits the tail of another snake, the damaged portion of the tail is deleted. Once a snake has grown a long enough tail, goal posts will appear at the bottom of the screen. Navigating the correctly colored snake through the goal posts represents completion of one third of the puzzle. Navigating all three Snakes through the goal posts represents completion of the game. Specially-colored pellets will appear randomly on screen for the snakes to collect. A snake that exits the edge of the screen will reappear on the opposing side of the screen in a wrap-around style.
- Requirement: The Snake game shall be considered complete once all three snakes have exited through the goal posts.
  - Implementation: The Wall computer will then proceed to the following chapter such as the Hyperspace game.

Hyperspace Game

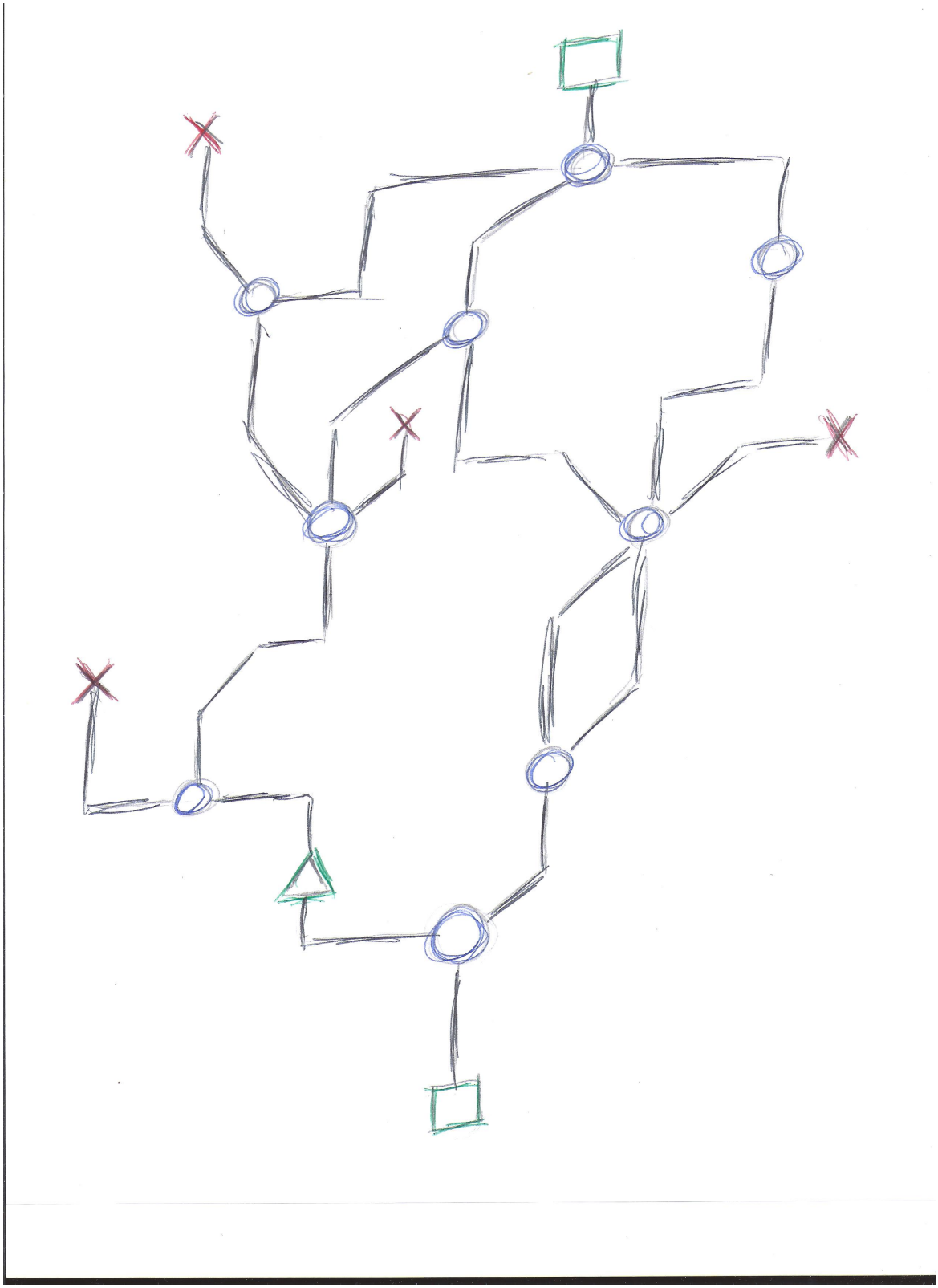
- Implementation: The Hyperspace game will start with a cut scene depicting the pod descending from the space craft and entering hyperspace. This cut scene will only be played once during a normal Hyperspace game play-through, regardless of the number of player deaths.



- Implementation: The Hyperspace game will consist of a cargo pod navigating through a series of hyperspace rings.



- Implementation: The console monitor will display a map of the hyperspace maze.

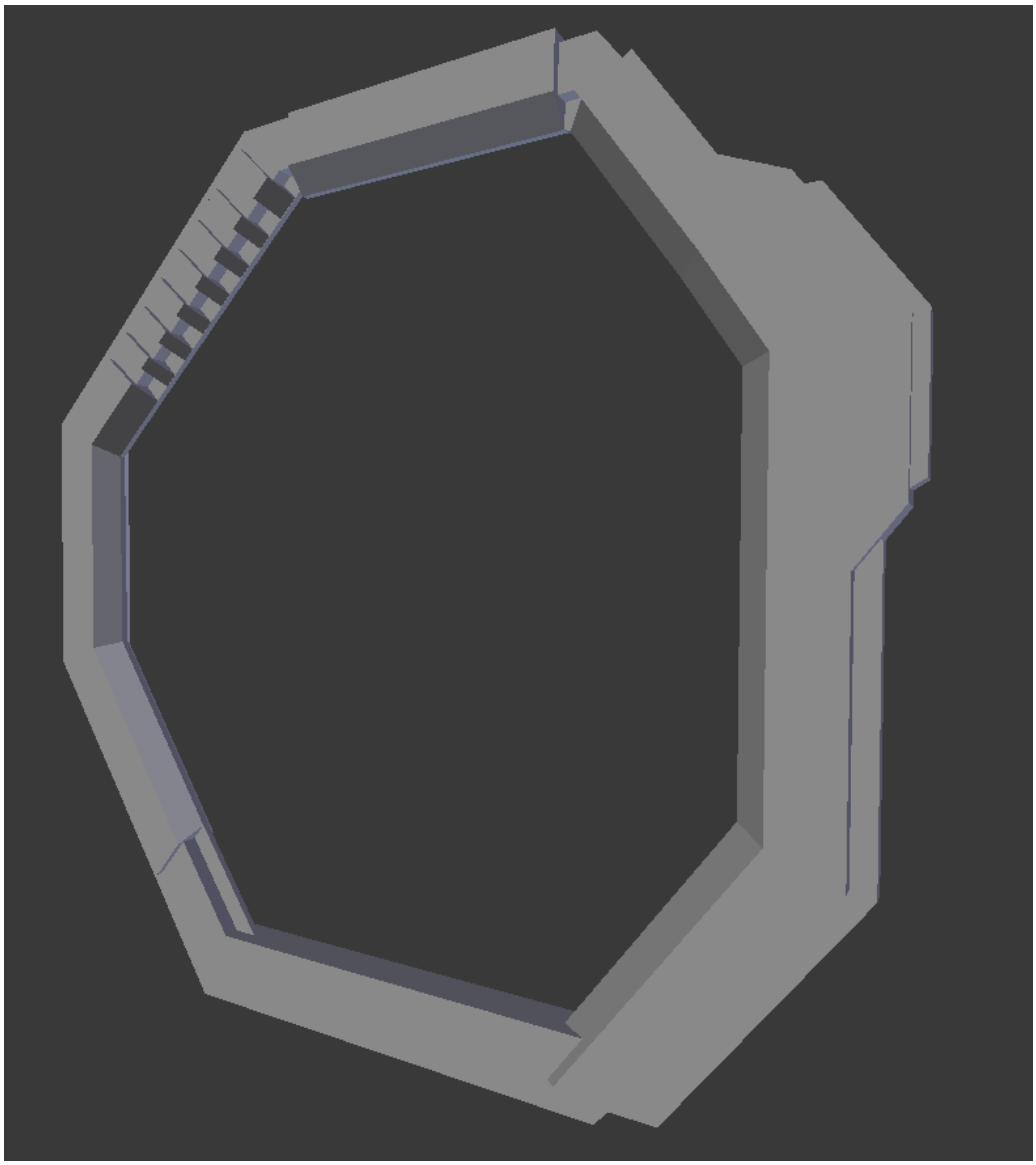


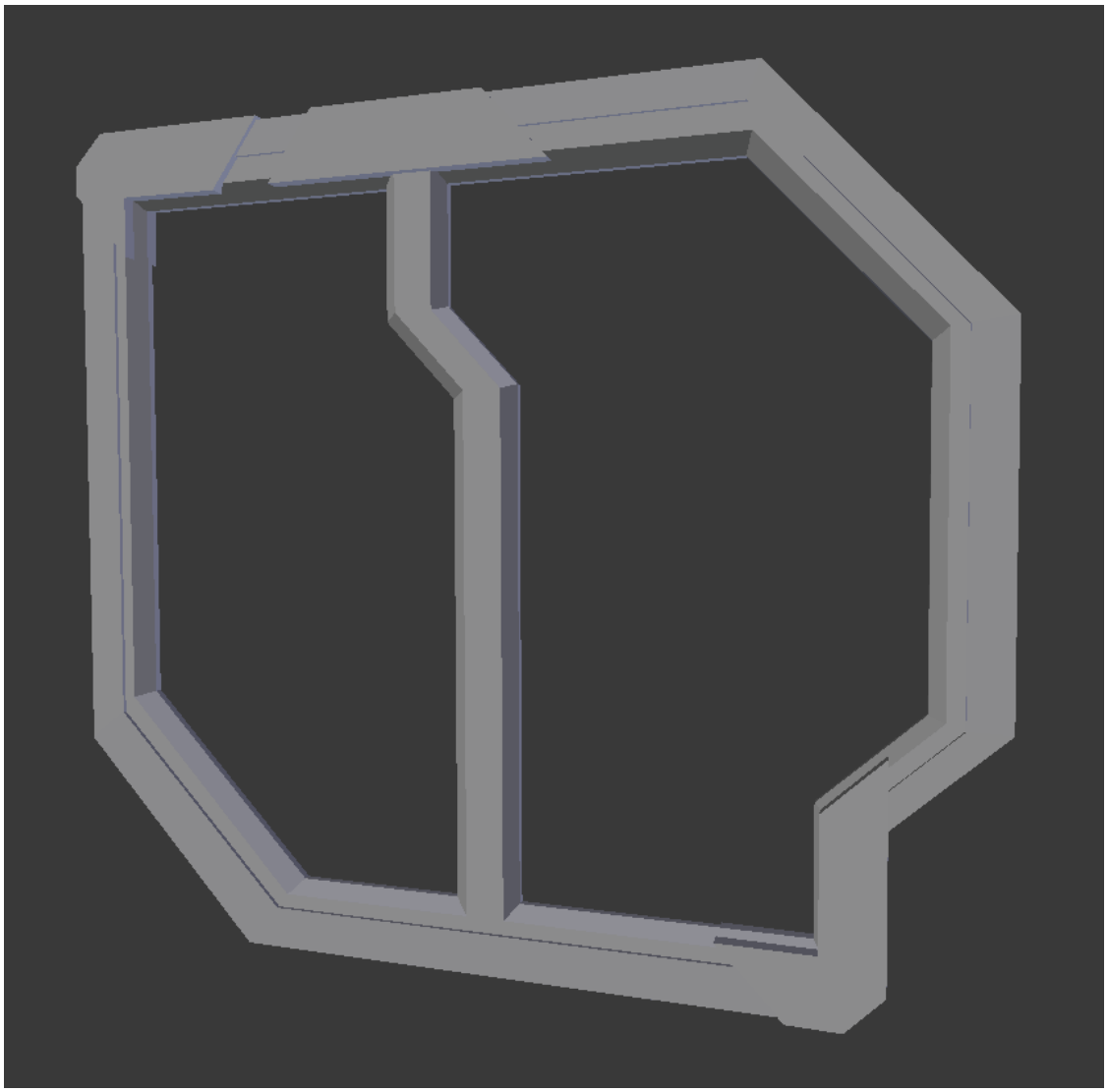
- Implementation: The music for this section will be inspired by the following reference songs:
  - Zabutom - Final Blast
  - Them's Fightin' Herds - Title Theme

    - Riggsmeister - To The Future

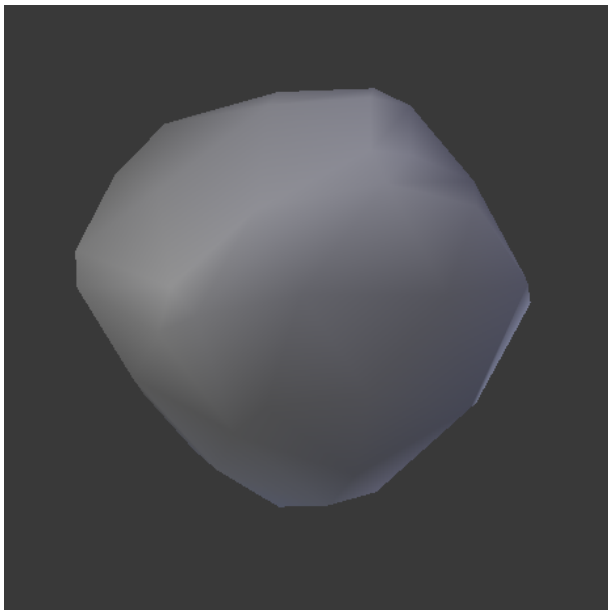
- Implementation: Sound effects will be played in response to user inputs in the Hyperspace puzzle, for example thruster and laser-firing sounds.
- Requirement: A thematic protective shroud shall enclose the speakers to protect them from players and hide them from view without degrading audio output.
- Requirement: The Hyperspace Game shall be designed to be completed in no more than 10 minutes, but allow for shorter or longer play times depending on player skill level.
- Requirement: Upon start of the Hyperspace game, the wall monitor shall emit a trigger signal to the console computer to transition to the map display.
- Implementation: The player character is a single cargo pod with thrusters and a laser.

- Implementation: The playing field consists of a branching hyperspace path filled with asteroid obstacles. The map depicts the full Hyperspace maze. The Hyperspace path is indicated via linear series of rings and occasional forks where players need to navigate to either the left or right side of the screen to select a branch.





- Implementation: The asteroids consist of three sizes: small, medium, and large. The medium asteroid can either be shot with the laser or avoided by moving the pod out of a collision course. The large asteroid has no room to maneuver around and must be shot with the laser. The small asteroid (a debris cloud) cannot be shot with the laser and must be avoided.



- Implementation: The four buttons in the console and four stomp pads act as an Ouija board for controlling the position of the pod. If only the only user input was the right button in the console, the pod would move at one-third speed to the right. If the only user input were the right stomp pad, the pod would again move at one-third speed to the right. When both the right pad and right button are activated synchronously, the pod moves at full speed to the right. The logic for the remaining three cardinal directions act in the same combinational manner. The joystick in the captain's chair controls the laser cross hairs. When the fire button is pressed, a laser fires. If the laser hits a medium or large asteroid the asteroid will be destroyed. Targeting a small asteroid debris field with the laser does nothing.
- Implementation: Hitting an asteroid triggers a player death. The pod is reverted back to the start of the hyperspace map and briefly flashes to indicate a new life.

Proctor

- Requirement: The proctor shall emit signal(s) over Ethernet to prompt the wall and console computers out of idle standby and begin the 60 minute countdown timer.
- Requirement: The proctor shall have the capability to change the remaining time to complete the room.
- Requirement: The proctor shall have the capability to independently advance from one puzzle to the next.
- Requirement: The proctor shall have the ability to view the status of the available puzzles.
    - Requirement: The proctor shall have the ability to view the players within the room.
        - Requirement: The proctor shall receive no fewer than one snapshot of the room at no less than once per second.
    - Implementation: Allow the state of the finale computers to be queried and configured through an Ethernet connection. The computers will be organized as a two-tiered state machine with a top level 'book' controlling the narrative sequence of 'chapters'. Each chapter is a game, puzzle, video sequence, or blank screen.
- Requirement: The goal shall be to give the proctor the ability to pause and resume the countdown timer and puzzle progression manually.
- Requirement: The proctor shall have the capability to broadcast verbal hints to players without using the finale speakers.

Credits

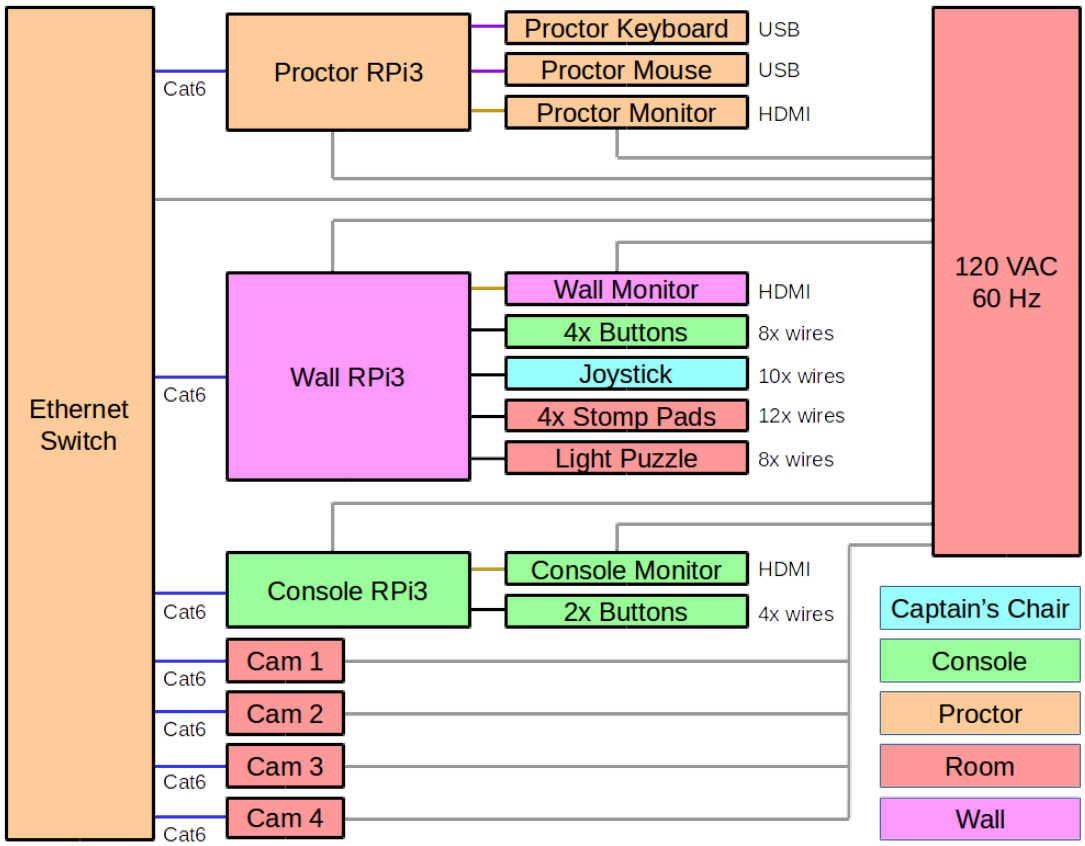
-

Implementation: The credits sequence will be implemented as a slide show. Following a successful room completion in under 60 minutes, half the screen will display credits for the puzzle builders, artists and support staff. The other half of the screen will display original static artworks of pod exiting hyperspace, entering a planet's atmosphere, descending under parachute, landing on the surface, and astronauts removing supplies from the pod. If the room is not completed in under 60 minutes, the artwork will be replaced with detailed pictures of the puzzles to highlight the assembly and design work.

- Implementation: The music for this section will be inspired by the following reference songs:
  - Incredible Machine 3 - Unplugged
  - Firewatch - Camp Approach
- Requirement: When the puzzle is successfully completed in under 60 minutes, the total time taken to solve the game shall be displayed on screen during the credits sequence.
- Requirement: The wall computer shall play a pre-scripted credits sequence for no longer than 60 seconds following the completion of the Hyperspace puzzle.
- Implementation: Following the credits sequence, a brief cut scene showing the corporate logo will play.
- Requirement: If the room was successfully completed in under 60 minutes, a humorous blooper cut scene shall play after the corporate logo as a reward to the players for successfully completing the room.
- Implementation: Following either the blooper reel in a successful room completion, or the corporate logo in a failed room play-through, the wall monitor shall proceed to the idle standby state.

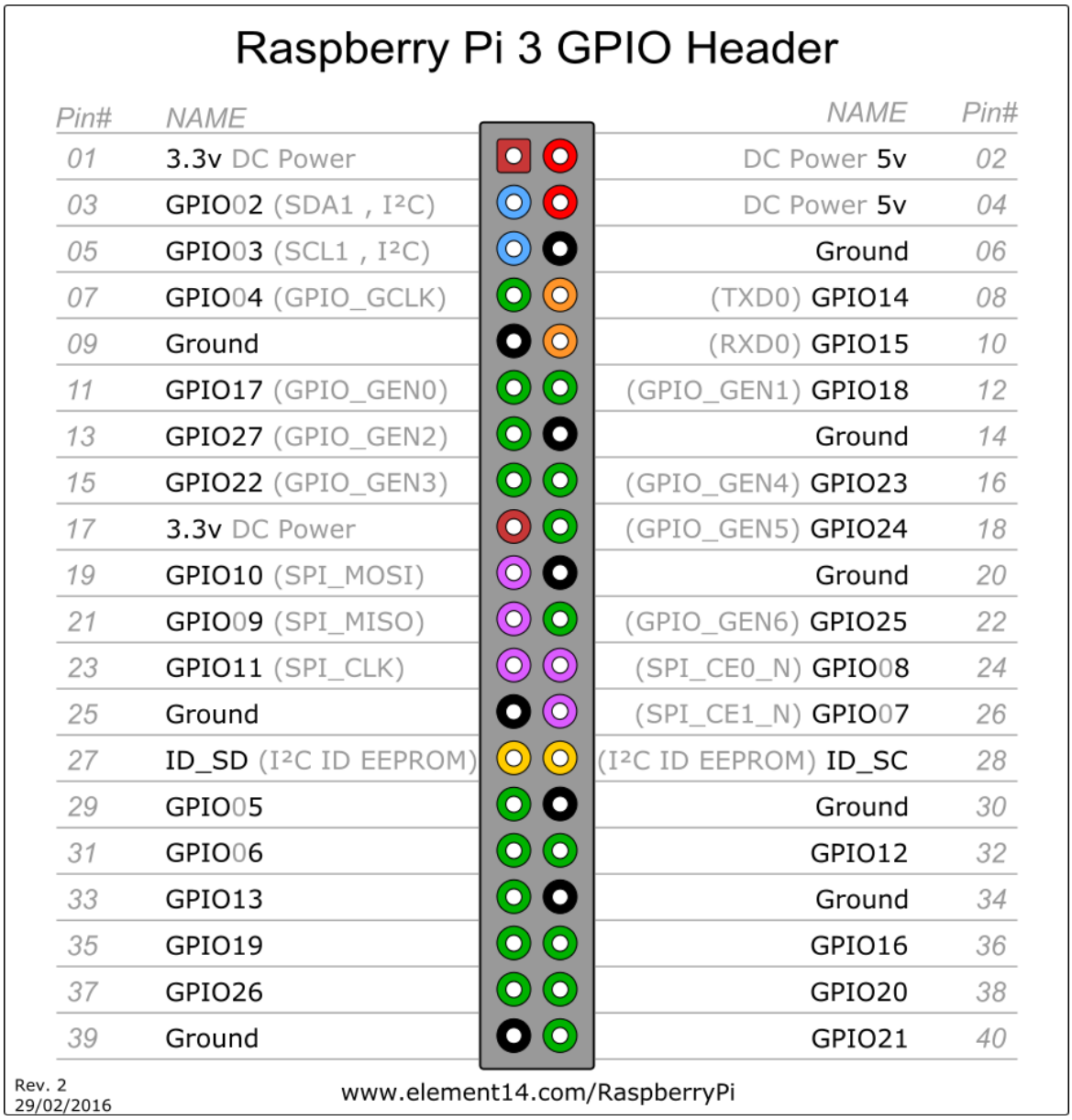
Detailed Implementation

Connection Diagram



Wall Computer

- Implementation: The Wall PC will automatically proceed through these chapters in the listed order as players play the finale unless overridden by the proctor
  - Idle Standby: The monitor will display a black screen and emit no sound. This chapter will complete when the proctor provides the appropriate input.
  - Opening cut scene: A cut scene explaining the room setting and objectives. The end of the chapter constitutes completion of the chapter.
  - Light puzzle: Display light puzzle component status. Display and update countdown timer. Activation of all four light puzzle components constitutes compeltion of the puzzle.
  - Instructions cut scene: A cut scene explaining the remaining objectives and user controls.
  - Snake: A 5-person video game where 1-2 players each control an on-screen character. The game is complete when all snakes are long enough and have exited the playing field.
  - Hyperspace: A 5-person video game where players work together to navigate a cargo pod through hyperspace, while avoiding and blasting obstacles. Reaching the end of the hyperspace maze constitutes completion of the game.
  - Credits: A pre-scripted sequence highlighing the efforts of the escape room development team.
  - Corporate logo cut scene: The corporate logo animation will play for a few seconds.
  - Blooper video: When the room is successfully completed in under 60 minutes, a bonus video will play after the credits, otherwise this chapter is skipped. The next chapter after this is to return to idle standby.
- Unaccessible Chapters: the following chapters will exist and can only be reached through proctor commands
  - Debug: Display the state of user inputs to aid in troubleshooting.
  - Pause: Hold count down timer, display a pause icon on the screen.
- Implementation: Pinout


Raspberry Pi 3 GPIO Header
Pin# NAME NAME Pin#
01 3.3v DC Power DC Power 5v 02
03 GPIO02 (SDA1 , I²C) DC Power 5v 04
05 GPIO03 (SCL1 , I²C) Ground 06
07 GPIO04 (GPIO_GCLK) (TXD0) GPIO14 08
09 Ground (RXD0) GPIO15 10
11 GPIO17 (GPIO_GEN0) (GPIO_GEN1) GPIO18 12
13 GPIO27 (GPIO_GEN2) Ground 14
15 GPIO22 (GPIO_GEN3) (GPIO_GEN4) GPIO23 16
17 3.3v DC Power (GPIO_GEN5) GPIO24 18
19 GPIO10 (SPI_MOSI) Ground 20
21 GPIO09 (SPI_MISO) (GPIO_GEN6) GPIO25 22
23 GPIO11 (SPI_CLK) (SPI_CE0_N) GPIO08 24
25 Ground (SPI_CE1_N) GPIO07 26
27 ID_SD (I²C ID EEPROM) (I²C ID EEPROM) ID_SC 28
29 GPIO05 Ground 30
31 GPIO06 GPIO12 32
33 GPIO13 Ground 34
35 GPIO19 GPIO16 36
37 GPIO26 GPIO20 38
39 Ground GPIO21 40
Rev. 2
29/02/2016
www.element14.com/RaspberryPi

TODO UPDATE

Sheet1

BCM	wPi	Name	Physical	AD/DA	Motor_MC33886	Joystick	Button
-	-	3.3V	1	3.3V			
2	8	SDA.1	3	NC			
3	9	SCL.1	5	NC			
4	7	GPIO.7	7	NC		FIRE	
-	-	0V	9	GND		GND_FIRE	
17	0	GPIO.0	11	DRDY			
27	2	GPIO.2	13	PDWN			
22	3	GPIO.3	15	CS0_ADS1256			
-	-	3.3V	17	3.3V			
10	12	MOSI	19	DIN			
9	13	MISO	21	DOUT			
11	14	SCLK	23	SCK			
-	-	0V	25	GND		GND_1	
0	30	SDA.0	27	NC			
5	21	GPIO.21	29	NC		JOY_1	
6	22	GPIO.22	31	GPIO	M3_BLUE		
13	23	GPIO.23	33	GPIO	M4		
19	24	GPIO.24	35	GPIO			BTN_1
26	25	GPIO.25	37	GPIO	PWMA		
-	-	0V	39	GND			GND_1
-	-	5V	2	5V			
-	-	5V	4	5V			
-	-	0V	6	GND			
14	15	TxD	8	NC			
15	16	RxD	10	NC			
18	1	GPIO.1	12	RESET			
-	-	0V	14	GND		GND_2	
23	4	GPIO.4	16	CS1_DAC8552			
24	5	GPIO.5	18	NC		JOY_2	
-	-	0V	20	GND		GND_3	
25	6	GPIO.6	22	NC		JOY_3	
8	10	CEO	24	NC			
7	11	CE1	26	NC			
1	31	SCL.0	28	NC			
-	-	0V	30	GND			
12	26	GPIO.26	32	NC	PWMB		
-	-	0V	34	GND		GND_4	
16	27	GPIO.27	36	NC		JOY_4	
20	28	GPIO.28	38	NC	M1_RED		
21	29	GPIO.29	40	NC	M2_GREEN		

Page 1

Console Computer

- Implementation: The Console will have references to the following chapters but will only enter/exit chapters in response to external (Wall PC) commanding
  - Idle Standby: The monitor will display a black screen.
  - Light Puzzle Standby: The monitor will display a black screen.
  - Morse Code: The puzzle will accept user input via 'dot' and 'dash' buttons and display information on the screen

when the proper passcode is entered.
    - Snake Game Standby: The monitor will display a black screen.
    - Map: The monitor will display the hyperspace map and the location of the cargo pod
    - Credits Standby: The monitor will display a black screen
- Unaccessible Chapters: the following chapters will exist and can only be reached through proctor commands
    - Debug: Display the state of user inputs to aid in troubleshooting.
    - Pause: Hold count down timer, display a pause icon on the screen.
- Implementation: Pinout

TODO CREATE

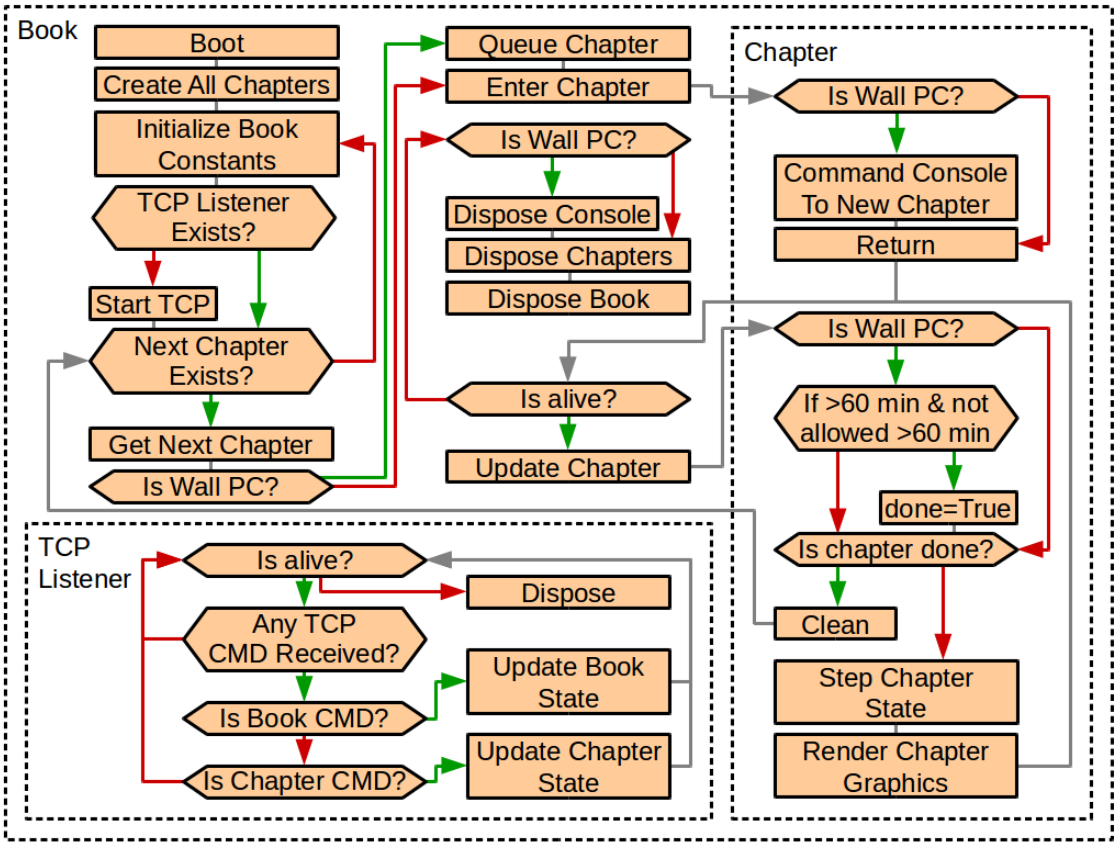
State Flow Diagram

The Wall computer is treated as a MASTER to the SLAVE Console. The Proctor computer is treated as a MASTER to the SLAVE Wall.

The following is a state transition diagram that will have nearly-identical copies running on both the Wall and Console computers.

- Book
    - **Boot:** The Raspberry Pi will perform a Linux boot sequence followed by execution of the main Python program
    - **Create All Chapters:** The object-oriented chapters are created, but not executed here. Execution of time-intensive processes such as loading video codecs and 3D graphics will also occur here. Following Chapter creation, the chapter's clean method will be called to configure the chapter constants to default values.
    - **Start TCP:** A separate thread is launched to listen for TCP commands from the proctor console. The TCP Listener is created after the constants have been defined to avoid the potential for the proctor to access constants before they exist.
    - **Next Chapter Exists:** Fetches the next chapter queued in the book state space. For example if the current puzzle is "Snake", the next chapter will be "Hyperspace".
    - **Queue Chapter:** If the PC this code is running in is the Wall (MASTER), then the "next chapter" is immediately defined.
        - For example in the Wal computer, if the upcoming chapter is "Hyperspace" the queued chapter will be "Credits". Queuing the following chapter immediately upon entering the current chapter allows the proctor to configure the next chapter at any time. For example, the proctor could set "Snake" as the next chapter while "Hyperspace" is running. Following a "next chapter" command by a "chapter done" command will transition the state machine to that commanded chapter.
        - If the PC is the Console, the next chapter is not updated. The Console is run as a SLAVE to the Wall.
    - **Enter Chapter:** After fetching the chapter to run, a chapter configuration sequence is run.
    - **Is alive?** If the proctor has sent a "dispose" command, the is_alive state will be False, prompting the Wall PC to send a command to the Console to die, followed by disposing of the Wall's own chapters and book. This ensures a clean exit to the desktop when commanded by the proctor.
    - **Update Chapter:** The Book calls the Chapter update loop at nominally 30 Hz. This loop updates the Chapter state machine and renders graphics on the screen.
- TCP Listener
    - **Is alive, TCP Command Received?** The TCP listener listens for new commands in an infinite loop at nominally 30 Hz.
    - **Is Book Command?** If the proctor command is formatted as a book command, it is processed. This includes the "dispose" command which will stop both the TCP Listener and the Book threads. "set next chapter" is another book command that the proctor can send.
    - **Is Chapter Command?** If the proctor command is formated as a chapter command, it is processed. This includes the "done" command which stops the execution of the current chapter and prompts the Book state machine to proceed to the next chapter.
- Chapter

- **Enter Chapter:** Every time the chapter is run, the first method call is a dedicated initialization sequence. For the Wall PC, an important step here is to send the TCP commands to ensure the Console is in the proper state.
- **Update Chapter:** Some chapters like Hyperspace or the Light Puzzle need to stop operations if the 60 minute window expires. This update step checks whether the 60 minutes has expired and if so, exits the chapter. Other chapters like the Credits will always play, so this check is ignored. The game state is incremented and a graphical frame is displayed.
- **Clean:** This method will always be called prior to exiting any chapter. Configuring the constants on exit in preparation for the next run, rather than on initialization immediately before running chapter, allows the proctor time to configure constants while running other puzzles. This ensures a clean entry into the chapter in whatever state the proctor desires. This avoids the design limitation of requiring the proctor to change the chapter state only after entering the puzzle, which may break the illusion for players as graphics and game state rapidly change on screen. This clean step is also where resource deallocations will occur such as stopping playback of video.



Schedule

The following list of tasks shows the planned date of completion of significant code elements to achieve the mid-March "95% completion" deadline. A functional build entails displaying relevant content on the monitors in response to user and proctor inputs to facilitate an end-to-end play-through of the room.

- Dec 18, 2017 Architecture Plan
- Dec 31, 2017 Functional Architecture, IO Handler, Core
- Jan 2, 2018 Hand-off to Morse Code puzzle developer
- Jan 8, 2018 Video Functional Build
- Feb 5, 2018 Hyperspace Functional Build
- Feb 19, 2018 Snake Functional Build

- Feb 26, 2018 Credits Functional Build
- Mar 12, 2018 Integration with Morse Code