Space Escape Room - TDD

27.07.2017 - PRESENT

Written by: Joel Gabriel

Stray Pixel Studios™

Advanced Diploma of Professional Games Development - Programming

Meet the team:

Zachary Farmer (**Designer**), Luccas Hobbs (**Designer**), Dion Tomholt - (**Lead Artist**), Aaron Watson (**Artist**), Joel Gabriel (**Lead Programmer**), Matthew Jones (**Programmer**)

TABLE OF CONTENTS

Game Overview

A VR game where the player must solve puzzles using a variety of tools at his disposal, in order to repair a damaged space ship and reactivate the life support which in turn, restores the oxygen levels, saving your life.

Engine

Unity3D v5.5.1:

The game engine we are using for this project is the unity engine. This was chosen due to it being 100% free and also the most familiar engine amongst our team mates. It also now features in-engine version control that is simple to use.

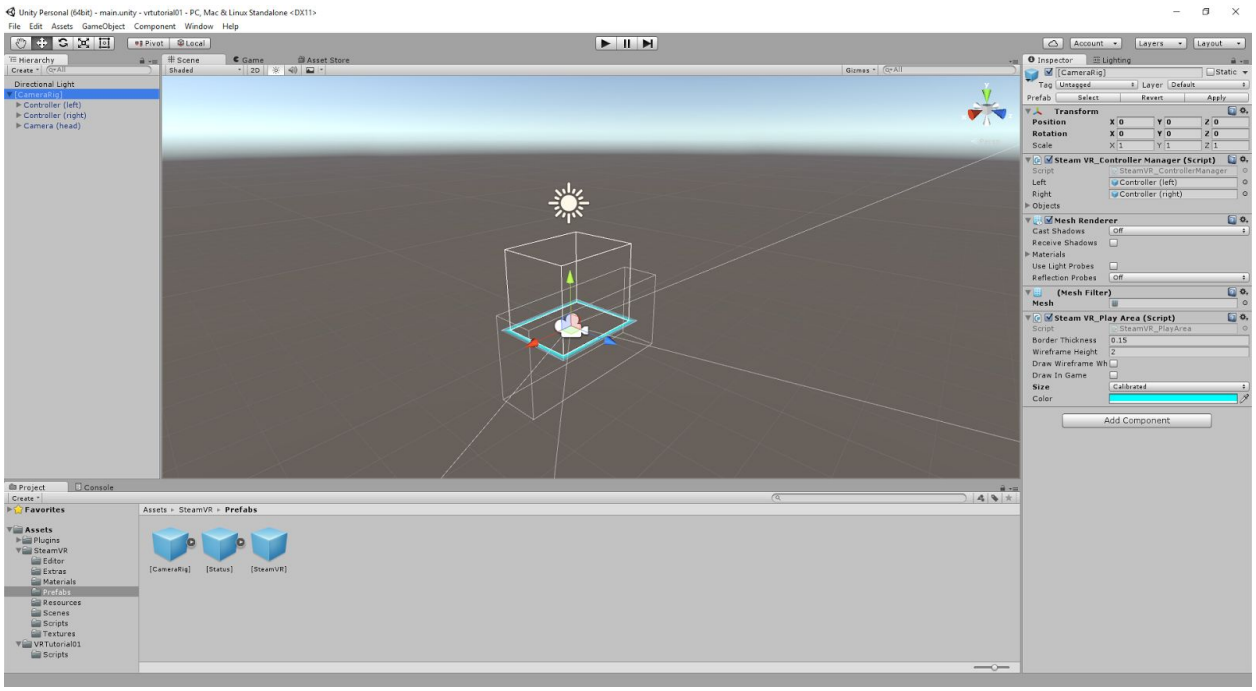
Plugins/External Tools:

SteamVR:

steamVR is an essential component to the VR development process and handles all logic that lets unity communicate between steamVR and the game inside the editor, it also provides you with a camera rig prefab to get you started quickly

NewtonVR:

NewtonVR is a free plugin on the asset store that handles physics interactions in VR. It uses its own custom logic to extend off of Unity's physics, to create very realistic interactions.



System Requirements

Hardware Requirements:

Desktop PC / Laptop:

The specs below take into consideration the htc vive hardware requirements.

Minimum PC specs

Processor: Intel Core i5-4590 or AMD FX 8350 (equivalent or better)
Graphics card: NVIDIA GTX 970 or AMD Radeon R9 290 (equivalent or better)
RAM: 4GB or more.
Video output: One HDMI 1.4 or one DisplayPort 1.2.
USB: One USB 2.0.
OS: Windows 7 or newer.

Desired PC specs:

Processor: Intel Core i5-4590 or AMD FX 8350 (equivalent or better)

Graphics card: NVIDIA GTX 1060 or AMD Radeon RX 480 (equivalent or better)

RAM: 4GB or more

Video output: HDMI 1.4 or DisplayPort 1.2

USB: One USB 2.0

OS: Windows 7 or newer

HTC Vive Kit:

This is required to play this game as it is a VR game. It will need to be setup as the setup guide for htc vive directs you, in order to run properly.

Software Requirements:

Unity 3D:

version 5.5.1 is required for the development of this project, and it is important that everyone is on this version of unity in order to prevent issues.

Visual Studio:

Visual studio is required for the code editing and debugging of scripts, this can also be done using monodevelop, which comes as a part of the unity engine, however it is suggested that everyone uses visual studio.

Steam/SteamVR:

SteamVR is essential both for the development of the game and to actually be able to run the game.

Source Tree:

SourceTree is required for every member of the team, and is what we will be using as version control.

Photoshop:

Photoshop is required by artists for the creation of textures, UI/GUI art, game icons etc.

Maya 2017:

Maya is required by the artists for creation of meshes, rigging and animation.

Zbrush:

Zbrush will be used by the artists for sculpting organic shapes and doing fine surface detail work.

Substance Painter:

Substance Painter will be used heavily in our texturing pipeline as well as used extensively for the baking process.

Player View

Viewpoint:

The viewpoint for this project will be 1st person due to the the nature/purpose of VR, putting the player in the eyes of the man in danger.

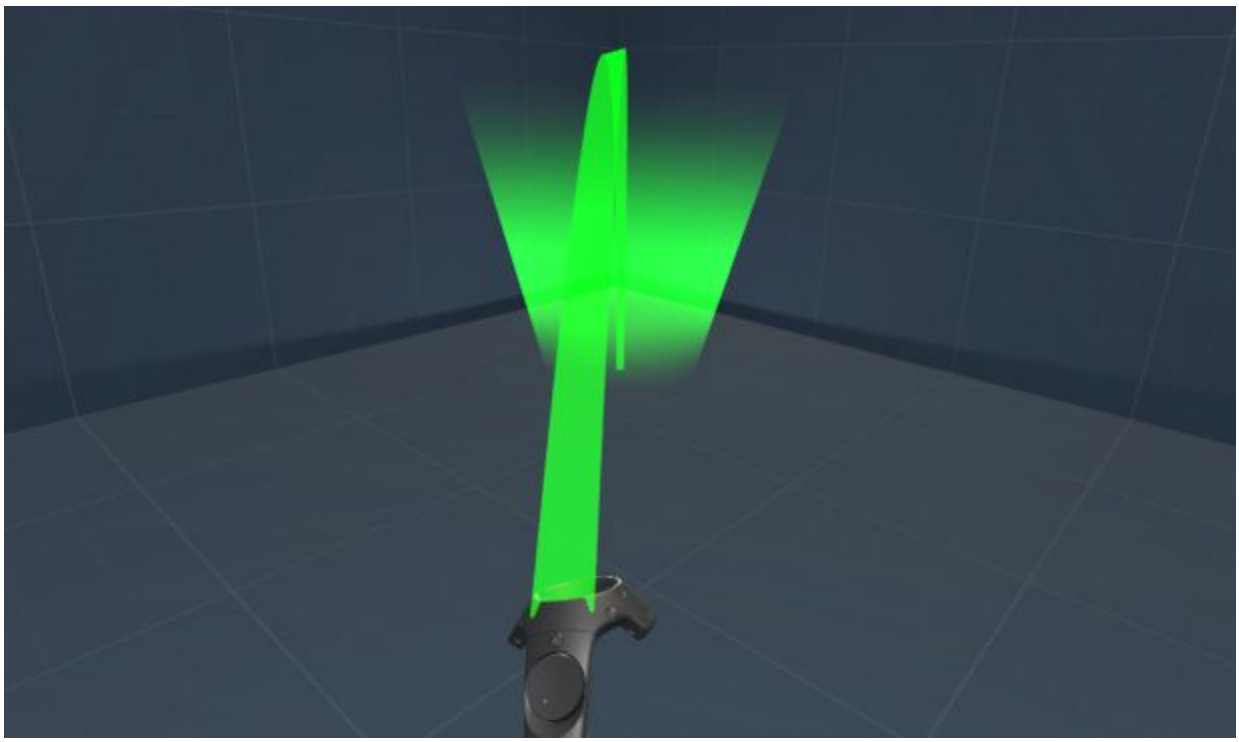
Camera:

The camera will be setup using the VR camera rig prefab provided by the steamVR plugin.

We will be using separate cameras that are used for the tablet devices camera in VR. This will then be projected onto a render texture, allowing users to take selfies and photos of key items.

Movement:

Movement will be handled by the ability to teleport around the damaged space ship. There is also a chance of a more natural locomotion technique being implemented.



Movement Restrictions:

The player can not teleport into zones that have not been unlocked, and cannot teleport through walls, or into places that are deemed ‘un- standable’.

Due to the fact a wall restriction in VR, does not correlate over to the physical world, this allows players to pass through walls if they teleport into a corner and walk forward in their physical play space, and their head will clip the wall.

However this is less immersion breaking/disorienting than merely stopping the camera in its tracks, which in turn induces motion sickness and disorients the player, so instead, **any key items behind walls or hidden in other rooms, won’t be rendered to the player, giving them no strategical advantage, nor motion sickness.**

Class List I

Main Menu Classes:

Our classes names will always precede with SE, meaning Space Escape. We will also be using m_ and a_ for member variables and arguments to keep a consistent coding style.

SE_ScreenState.cs	This class is required for the initial startup of the game, and handles the transition to the main game. It also has a quit option and displays the control scheme to the player via ui elements attached to the controllers themselves.
SE_MenuScreen.cs	This script is attached to the main menu part of the UI and will update once the UI component is enabled. It also handles the transition into the main game.
SE_OptionsScreen.cs	This script is attached to the options part of the UI and will update once the ui component is enabled.
SE_CreditsScreen.cs	This script is attached to the credits part of the UI and will update once the ui component is enables

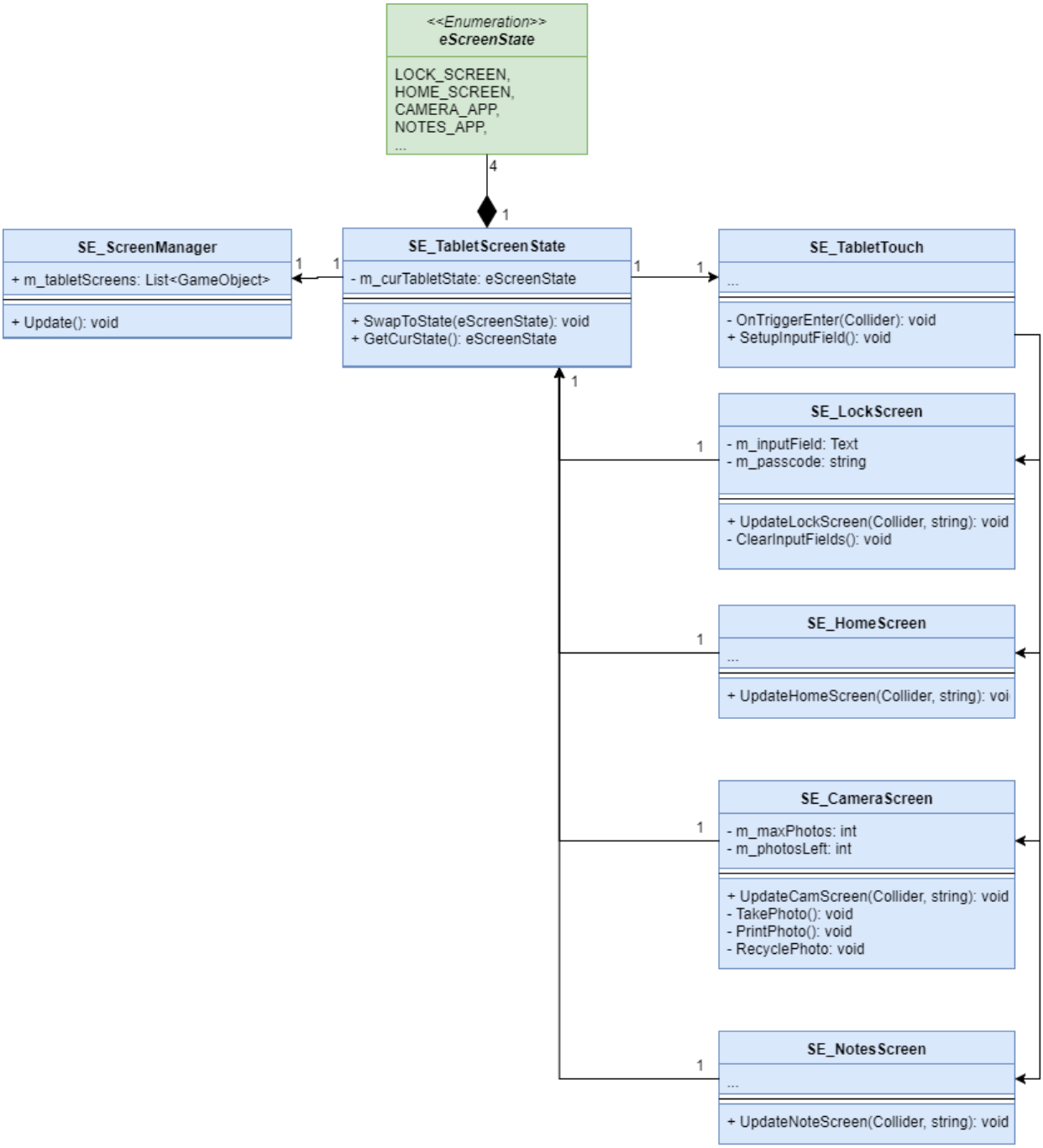
Class List II

Tablet Classes:

These classes handle all the in game, tablet interactions which include, touch screen with stylus, unlocking and locking and screen transitions.

SE_TabletScreenManager.cs	This class is required to update the visuals of the screen depending on the screen state. It handles the visuals and when to enable/disable certain screens and handles transitions.
SE_TabletScreenState.cs	This class is used to define the different states the screen can be in such as: *LOCK_SCREEN*, *HOME_SCREEN*, *CAMERA_APP*, *NOTES_APP*. And also has public methods that swap out these states.
SE_HomeScreen.cs	This class is used to handle the logic for the home screen itself, and determines what certain buttons do on the screen.
SE_LockScreen.cs	This class is used to handle the logic for the lock screen, and handles correct and incorrect passwords appropriately.
SE_CameraScreen.cs	This class is used to handle the logic for the camera screen, and correctly displays what the tablet camera is seeing, as well as prints out photos you take.
SE_NotesScreen.cs	This class is used to handle the logic for the notes screen, and allows the user to write and view certain notes.
SE_TabletTouch.cs	This class handles the events that take place when you touch the screen of the tablet with your stylus. Taking into consideration what screen is currently active.

Tablet Class Diagram:

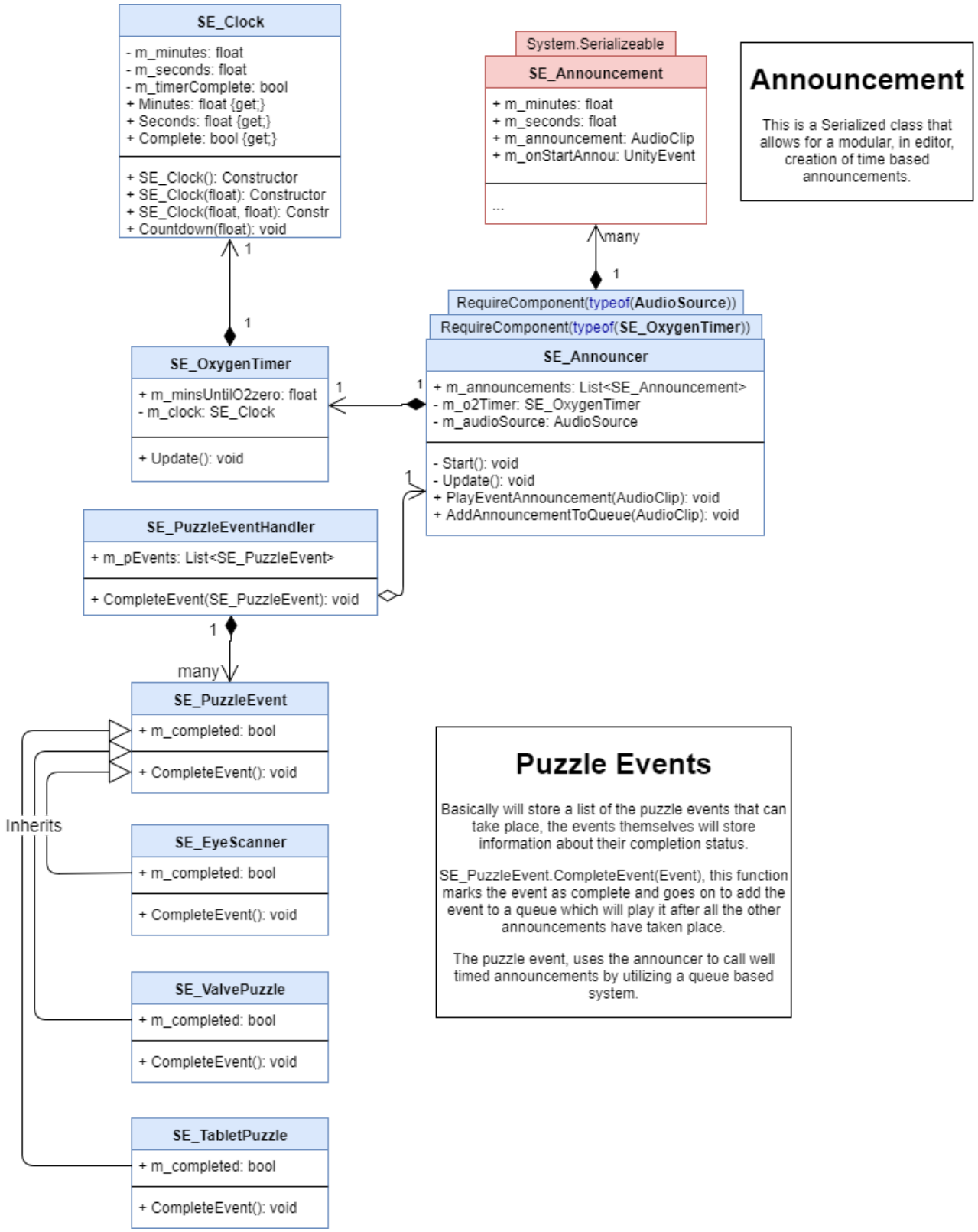


Class List III

Gameplay Classes:

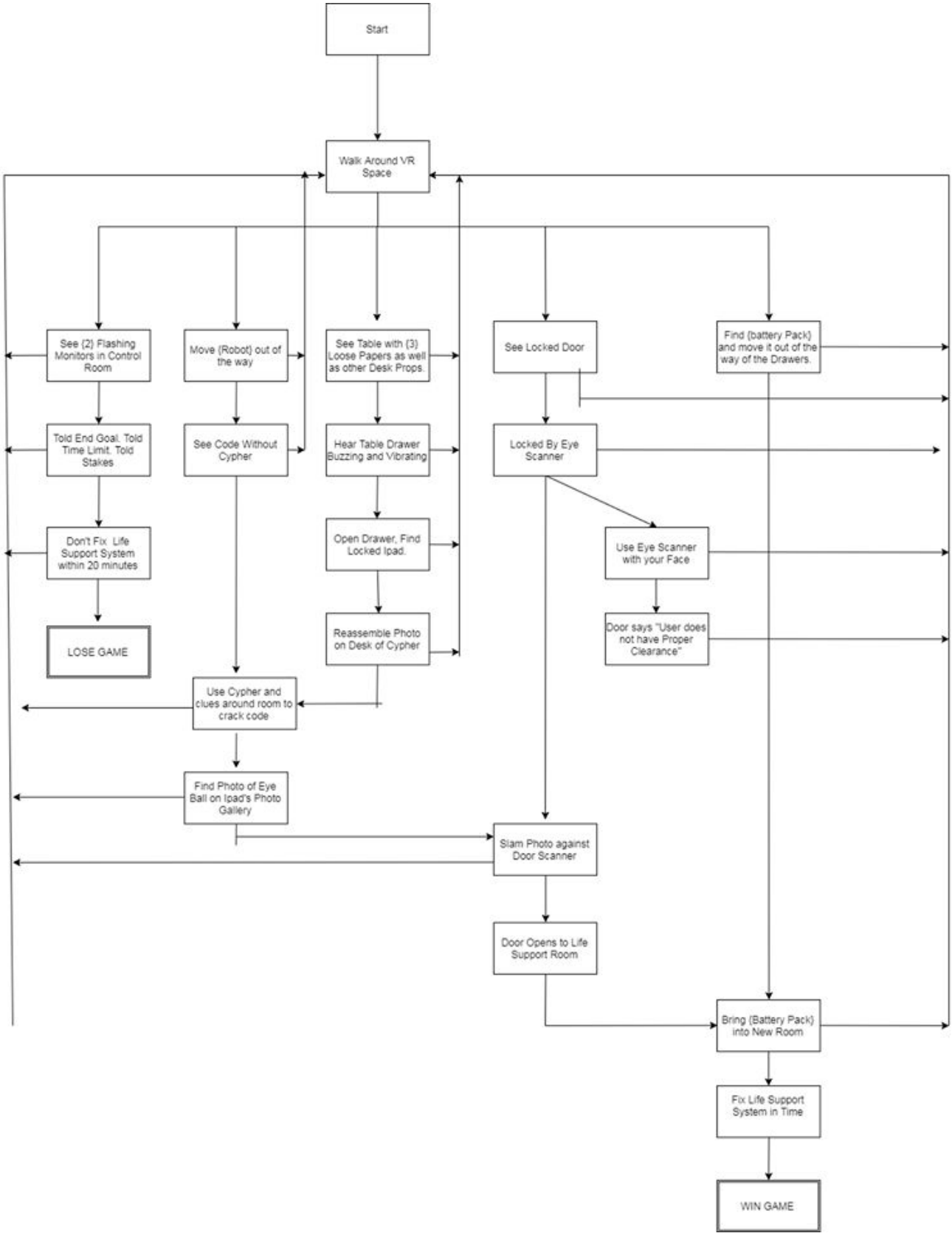
SE_PuzzleEvents.cs	This class is going to be keeping track of all the puzzle events and handling the events that take place after puzzle completion.
SE_EyeScanner.cs	This class handles the logic for the eye scanner, it raycasts from the scanner if your HMD is in range and determines whether or not they have correct authority to access.
SE_Clock.cs	This class is going to handle counting down time in minutes and seconds.
SE_OxygenTimer.cs	This class handles the logic for the oxygen timer countdown.
SE_Announcement.cs	This class is going to be used by the Announcer and is a simple object that holds information about the time in which it activates, what sound play, and what event to trigger.
SE_Announcer.cs	This class utilises the Oxygen timer to handle announcements from the 'ship', Set up in a modular system to allow designer to set up announcements as they see fit.
SE_LightHandler.cs	This class handles the logic for the warning light spinning at the beginning of the game, and updates scene lighting depending on events taking place.

Gameplay Class Diagram:



Room Overview

Room Flowchart:



Room Puzzle Events:

Main Door Locked By Eye Scanner:

Use the tablet to take a photo of the boss's face in order to gain access to the life support chambers.

Photo of Cypher On Desk:

Helps solve the cipher, doesn't mean anything earlier on to the player, however will when they find the hidden code.

Hidden Code Without Cypher:

Looks like gibberish to the player, makes them try and find a clue to solve this puzzle.

Hidden & Passcode Locked Tablet:

Tablet vibrates in the draw, buzzes to get players attention using 3D sound.

Paper Rearranging Puzzle:

No code logic required, merely based on the look of the paper and being able to re-arrange them in order to get the correct number combination.

Art Specifications

2D Assets:

Format .png

File Name:	Art Type:	Texture Resolution:
SE_[ART ASSET NAME]_UI	UI	256x256
SE_[ART_ASSET_NAME]_Texture	Texture	1024x1024
SE_[ART_ASSET_NAME]_AnimTexture	Animated Texture	128x128 * 9 (sprite sheet)

3D Assets:

File Name:	Art Type:	Poly Count [limit]
SE_[ART ASSET NAME]_Prop	Prop	1k - 2k
SE_[ART_ASSET_NAME]_Char	Character	10 - 15k
SE_[ART_ASSET_NAME]_Env	Environment	5k

Materials:

File Name:	Material Type:	Texture Resolution
SE_[ART ASSET NAME]_Diff	Diffuse	[Same as texture]
SE_[ART_ASSET_NAME]_Norm	Normal	[Same as texture]
SE_[ART_ASSET_NAME]_Met	Metallic	[Same as texture]
SE_[ART_ASSET_NAME]_AO	Ambient Occlusion	[Same as texture]

Art Pipeline:

Maya, Zbrush, Photoshop, Substance Painter, Substance Designer, Engine

Audio Specifications

Format:

All of our audio files are going to be in *.wav* format as it is the most efficient and optimized way to do audio in the Unity engine.

Naming Convention:

SE_[FileName]_SFX.wav

SE_[FileName]_Music.wav

Sound List:

Sound	*Type*	*Variants*	*Looping*	*Max Length*
Background_Ambient	Music	No	No	20 Mins
Background_Ambient_2	Music	No	No	20 Mins
Background_Ambient_3	Music	No	No	20 Mins
Spawn_Noise_1	SFX	No	No	1 secs
Item_Pickup_Metal	SFX	Pitch/Tone	No	0.5 secs
Item_Pickup_Wood	SFX	Pitch/Tone	No	0.5 secs
Item_Pickup_Plasic	SFX	Pitch/Tone	No	0.5 secs
Item_Pickup_Metal_2	SFX	Pitch/Tone	No	0.5 secs
Item_Pickup_Wood_2	SFX	Pitch/Tone	No	0.5 secs
Item_Pickup_Plasic_2	SFX	Pitch/Tone	No	0.5 secs
Rumble_Viabrate_1	SFX	no	Yes	5 secs
Item_Drop_Metal	SFX	Pitch/Tone	No	1 secs
Item_Drop_Plastic	SFX	Pitch/Tone	No	1 secs
Item_Drop_Wood	SFX	Pitch/Tone	No	1 secs
Item_Drop_Ground	SFX	Pitch/Tone	No	1 secs
Endgame_Health_Critical	Music	no	No	4 mins
Endgame_Running_OutOf_Time	SFX	no	No	5 secs
Vault_Turning	SFX	no	No	5 secs
Finger_Tap	SFX	Pitch/Tone	No	0.5 secs
Ipad_Camera_Open	SFX	No	No	1 secs
Ipad_Camera_TakeShot	SFX	No	No	1 secs
Ipad_Unlock	SFX	No	No	1 secs
Ipad_Exit	SFX	No	No	1 secs
Ipad_Button_Press	SFX	Pitch/Tone	No	0.5 secs
Access_Denied	SFX	No	No	1 secs
Access_Granted	SFX	No	No	1 secs
3D_Monitor_Countdown	SFX	No	Yes	1 secs
Drawer_Slide	SFX	No	Maybe	5 Secs
Dead_Robot_Buzz	SFX	No	Yes	20 secs
Door_Open	SFX	No	No	5 secs

Audio Sourcing:

Due to limited knowledge on audio creation, our team will be sourcing a lot of our sound effects and any music files that are used. These will all be royalty free or under the creative commons license.