dmu.ac.uk
DE MONTFORT UNIVERSITY
LEICESTER

# CTEC 3451

*Development Project*

# Under Lock and Key

*Three-dimensional puzzle game*

## Supervisor

*Dr. Jethro Shell*
*jethros@dmu.ac.uk*
*0116 2078520*

*Authored by*

**Adam Leonard Hubble**
*P17175774*
*P17175774@my365.dmu.ac.uk*

# Contents

# Figures

## Tables

# Introduction

## Background

Under Lock and Key is designed to deliver a player experience that is recognizable yet disparate from existing three-dimensional puzzle games. For which, gameplay intends to present players with mechanically-driven challenges that are unique and less predictable than the genres counterparts; which aims to sustain player interest through heightening cognition and conventions of suspense, aesthetically and mechanically. Significant to player interest, Under Lock and Key introduces the narrative of a young girl who is trapped within a housed-environment, which is owned and roamed by a torturous entity. The young girl is introduced as the player-controlled character, that attempts to evade the entity to an eventual escape; to progress, the player is required to interact with a linearly sequenced series of puzzles that exist within multiple interconnected rooms. The puzzles featured within each of the rooms are authored by the application of models, lighting, cameras, materials, audio, and user interfaces (UI), that are all bound by scripts[1].

Relating to the project contract, the game adopts the intended escape-the-room orientation of gameplay, that closely aspires to the movie series, Jigsaw [1], and the mobile game series, The Room [2]. Aspiring to said series intends to stimulate a slowed pace of gameplay, that encourages cognitive interactions with puzzle components throughout the game[2]. As a device of pacing gameplay, the application utilises audio dynamically, for conveying the narrative, addressing player engagement and for introducing jump scare sequences; this is authored by an array of interactive and transitional sound effects. When addressing the multi-cultural learning criterion of the game[3], it was compulsory that non-linguistic audio was applied, to prevent restraints on a player's ability to progress, if puzzles were to feature auditory developments.

In further correspondence to the series mentioned, the game environments are and were expected to be low-key lit and compact in regard to scale, and further resemble room-like settings, hence escape-the-room. Due to the frequent recurrence of specified techniques in the genre[4], the adaptation of said genre conventions aims to incite players with gameplay objectives and a sense of familiarity that derives from games played prior; aside from enforcing the sinister art style of the game.

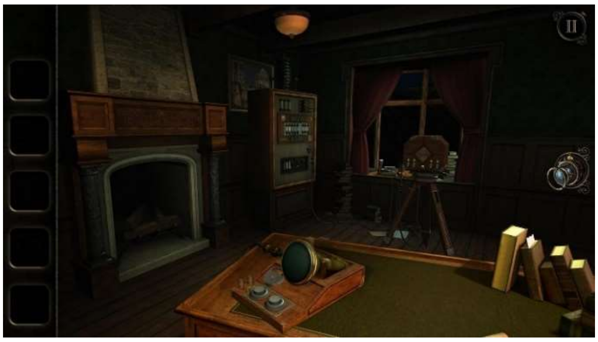


*Figure 2: The Room Three, low-key lit, compact, and room-like environment [3]*

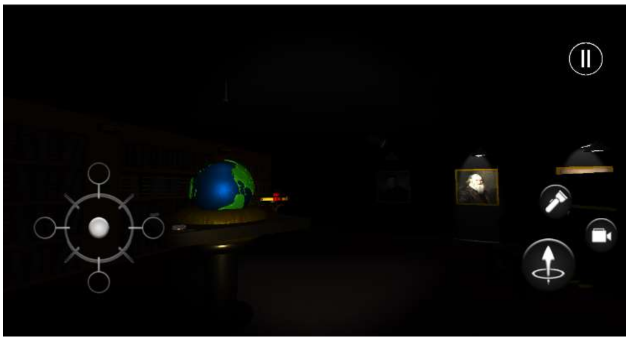


*Figure 1: Puzzle Room Two, low-key lit, compact, and room-like environment*

[1] These features are outlined in the project contract (see ***Appendix E***).
[2] These concepts are mentioned prior and within the literature review (see ***Appendix A***).
[3] This criterion was selected within the global checklist (see ***Appendix F***).
[4] As discovered from within the preliminary literature review (see ***Appendix A***).

Moreover, from a mechanical standpoint, puzzle components are and were typically noticed as being dependant of each other, for the purpose of becoming interactable, sequentially; said sequencing is controlled by the completion states of each puzzle room. Linearity can be acknowledged by the similarities between object geometry, object materials and the combination of sound effects, object animations and the presence of lighting within the game; these techniques are used as devices for offering players information, which encourage cognitive interactions with objects and gradual gameplay progression. As discussed within the literature review, information should be offered superficially; which aims to challenge the player, without instituting frustration from an inadequate amount of information provided.

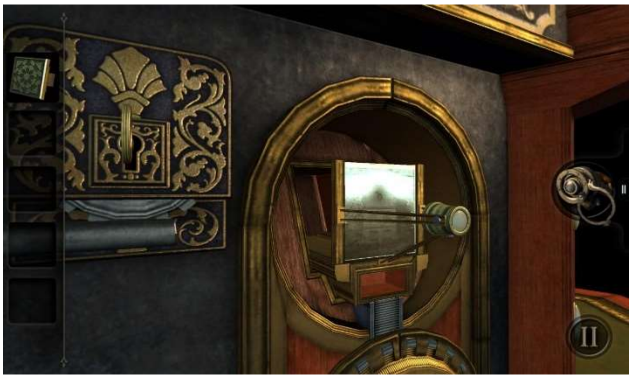


*Figure 4: The Room, item obtained modelled similarly to puzzle object [4]*

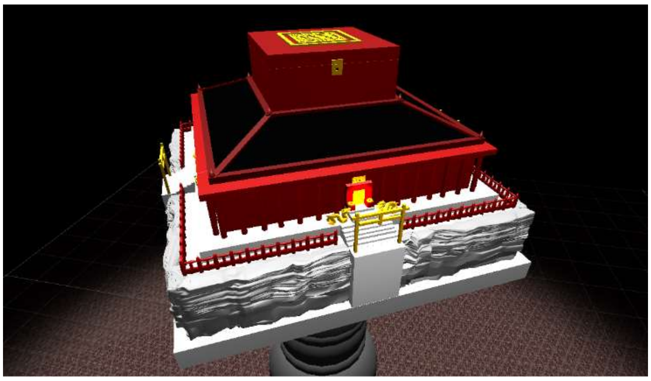


*Figure 3: Puzzle room two, music box item and music box stand modelled similarly*

As for the intended methods of interaction, puzzle interactions emphasise the conventional touch, swipe, and pinch related gestures; supported by both mobile and computer interfaces[5]. As discovered within the literature review, gesture interactions prevent the players viewport of the game world from being cluttered, as opposed to when there is a considerable presence of UI buttons; as well, interactions of this nature are viewed as more interesting, which sustains player interest and thereby adheres to the applications production purpose, to entertain. Being a conventional method of object interaction across the genre, expects to habituate players with the techniques of object interaction, from their experiences with interactive puzzler games prior[6].

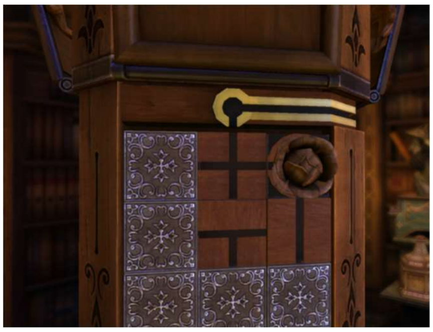


*Figure 7: The House of Da Vinci, multidirectional swipe gesture interaction, passage the flower through the tiles [7]*

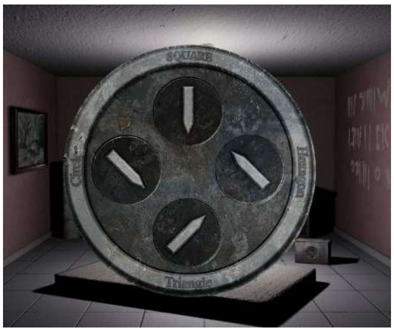


*Figure 6: Forever Lost, align the indicators, touch, and drag gesture interaction [8]*

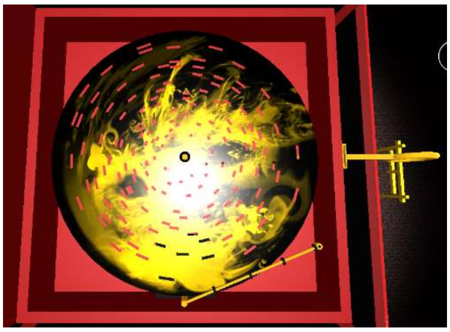


*Figure 5: Puzzle room two, music box disc needle alignment, pinch rotate left gesture interaction*

[5] Support for both platforms was necessary for the development, testing, and publication of the application.

[6] These interaction types are exemplified within The House of Da Vinci [5] and Forever Lost [6] mobile games.

## Motivation

As a student studying Computer Games Programming, the projects foundation as a game application was appropriate and well-tailored to demonstrating programmatical capability, given the game development experience accumulated over the duration of the programme. Furthermore, undertaking the development of a three-dimensional puzzler game with a first-person outlook on the game's environments, was advantageous for exercising my abilities to develop a game with a challenging nature, independently; one which showcases and prescribes personal development[7]. Moreover, developing a game as opposed to other software applications, was noticed to be more relevant to the programme of study and significant to my understanding, experience and practises of game development cycles and methodologies[8].

Relating to the criteria of the project undertaken, the three-dimensional, first-person basis for the production captivated my attention to the project proposal initially, due to my interests and regular interactions with mobile and console-based first-person shooter (FPS) titles[9]. From experience, said FPS titles often provided adrenaline-spiked experiences, when within close-ranged combat and scenarios that pressure solo efforts for achieving game-objectives; this enabled the interest and entertainment values of the games to be sustained, which became a design concept I had wanted to accomplish within a personal development.

*Figure 9: PUBG Mobile, first-person shooter, battle royale combat view [11]*

*Figure 8: Call of Duty: Mobile, first-person shooter, multiplayer game view [12]*

Moreover, from my past experiences with mobile puzzler games as a consumer, the demand for gesture interaction was of particular interest, as it presented differentiation from other genres of mobile game; which enabled the interest of the game to be engaged, prematurely. As well, many mobile puzzler games were recognized for incorporating narrative development, and an escape-the-room gameplay orientation that heightens player engagement; these posed as concepts for developing a game upon[10].

[7] Within three-dimensional modelling, visual motion graphics, image manipulation, sound editing, narrative building, programming, project management, research, and reporting.

[8] These are concepts applicated in real-world environments that surround the game development industry.

[9] Examples of such titles are Call of Duty: Mobile [9] and PUBG Mobile [10].

[10] Mobile game titles that captivate said interest are Adventure Escape [13] and Faraway 3: Artic Escape [14].

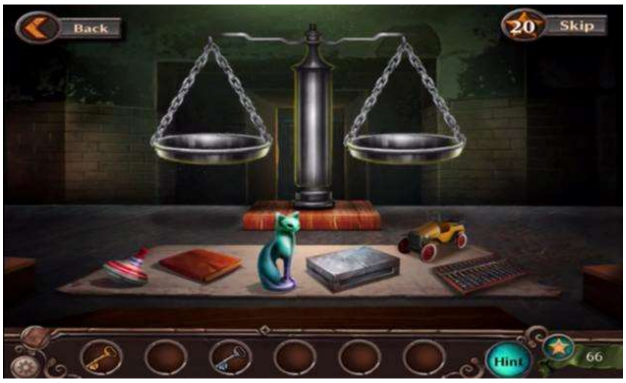


*Figure 11: Adventure Escape: Asylum, room-like environment and gesture endorsed puzzle interaction [15]*

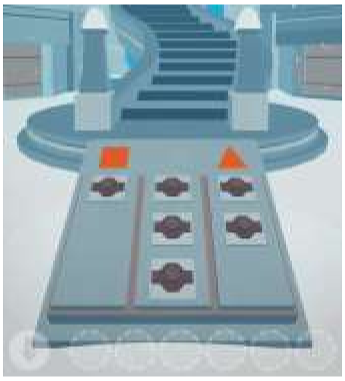


*Figure 10: Faraway 3: Artic Escape, area enclosed environment and gesture puzzle interaction [16]*

Also, as a procedure employed by my A-Level studies[11], an array of movie productions were examined for instituting the narrative development and mise-en-scene concepts of the game; this research practise was valuable for informing and reinforcing the aesthetic and mechanical design choices for the gameplay style projected. Conventional to the games studied, the movie productions that were also explored[12], exhibited many similarities aesthetically, auditorily and sequentially, through the narratives they uniquely depict. All of which discoveries, encouraged the emergence of the game's development initiatives, that have been adhered to throughout its development cycle.

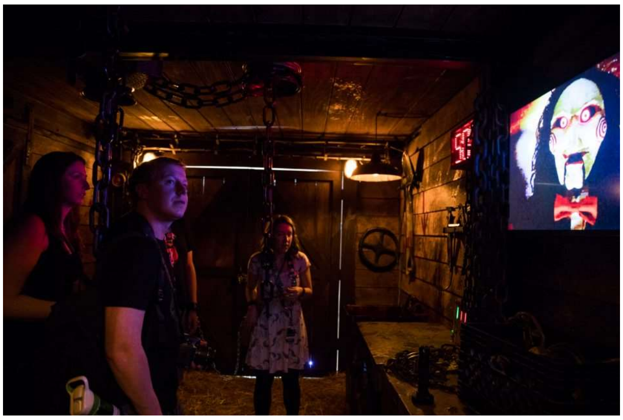


*Figure 12: Jigsaw, puzzle room aesthetic [18]*

*Figure 13: Escape Room, puzzle room aesthetic [19]*

## Aims and Objectives

For addressing the fundamental requirements of the software application, the deliverable state of the game should present players with a series of mechanically-driven puzzles[13]; each of which engage player interaction through the performance of gestures, that are expected to be compatibilized with device peripherals and interfaces available to them. In regard to the requirements of the puzzles implemented, players are expected to show basic numerical and observational understanding; these skills are significant to their progression throughout the game, and for personal development purposes, which should be addressed for all cultures and in all environments[14]. Lastly, the games scenes should feature multiple world spaces, for demonstrating the interconnectivity between the numerous puzzle rooms proposed.

[11] Media Studies.

[12] Jigsaw [1] and Escape Room [17].

[13] As specified within the project contract (see ***Appendix E***).

[14] This is detailed within the global checklist (see ***Appendix F***).

The undertaking of this project proposal aims to excel personal game development skills; focusing upon the simultaneous management and delivery, of a considerably complex piece of software, as a programmer and project manager. More so significant to the project's development, my ability as an newly educated object-orientated programmer, will be recognized; as will my ability to independently produce a game, that incorporates the proposed graphical elements, narrative development, and devices of emotional engagement[15].

# Body

## Project Specification

### Functional Requirements

#### Player Interaction

As an interactive puzzler game, implementing functionality that enables players to interact with objects in their environment, was compulsory to the game. As mentioned in prior sections of this document[16], players should be able to interact with objects throughout the game, using a series of touch, swipe, drag, pinch, and pinch rotate gestures. Interaction within the game could have been achieved with the prolific use of buttons, however, upon investigating existing titles within the three-dimensional puzzle genre [2] [5] [6] [13] [14], the application of gestures were appraised as a better alternative input technique; through gauging player interest and preserving the viewports that players are provided with, on the game world[17].

Relating to touch-based interaction, a player's ability to initiate object interaction and puzzle developments, has always been devised through casting rays at objects through the first-person camera's viewport; which visualises the screen space of a devices interface and player characters perspective. Object interaction through the first-person camera aims to enhance the realism, believability, and horror of the game, whilst reducing the complexity of camera mechanisms. To mobilise ray casting, it was required that interactable objects equip collider components, which are necessary in determining the object that intercepts the ray being cast.

[15] All of the features mentioned are extracurricular, self-disciplined principles of the game, which are more conventional to existing games, rather than what is expected from the programme.

[16] And within the functional requirements document (see ***Appendix B***).

[17] These concepts were introduced and addressed within the literature review (see ***Appendix A***).

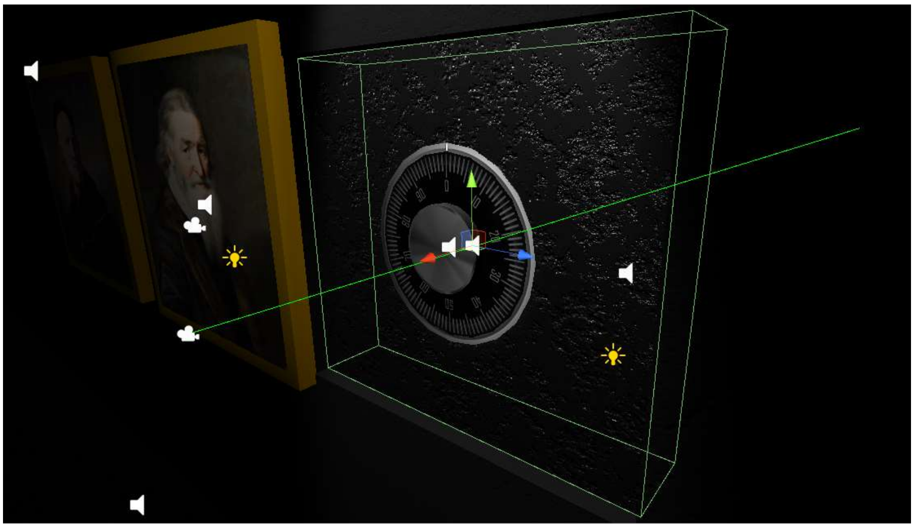


Figure 14: Ray casting, ray cast from the first-person camera frustum towards safe door object, safe door object equips collider component

In this determination, game object tags were implemented as a form of conditioning to easily identify and distinguish between the objects in each of the games scenes; this was especially beneficial when many game objects were interactable at one time, a scenario that exemplifies this is when interacting with an array of piano keys, as seen within the piano puzzle addressed in puzzle room two.

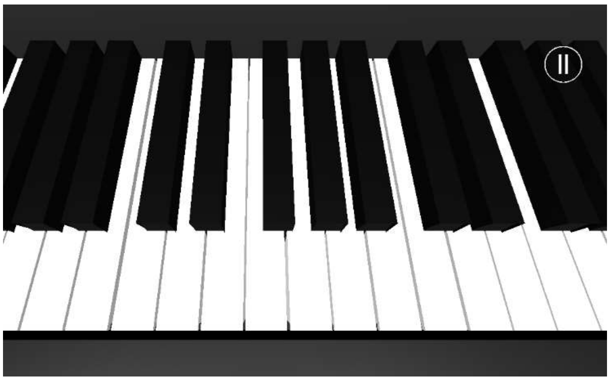
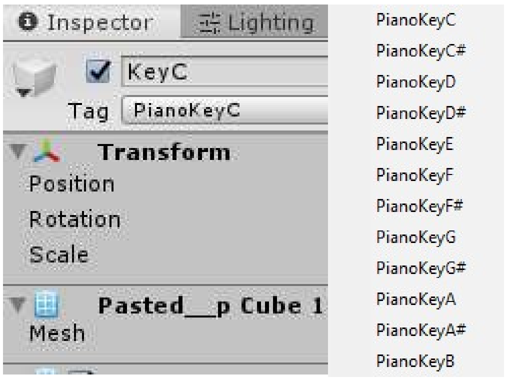


Figure 15: Puzzle room two, piano puzzle, inside piano key interaction view



☑ KeyC

Tag PianoKeyC

PianoKeyC
PianoKeyC#
PianoKeyD
PianoKeyD#
PianoKeyE
PianoKeyF
PianoKeyF#
PianoKeyG
PianoKeyG#
PianoKeyA
PianoKeyA#
PianoKeyB

Figure 16: Unity game engine inspector panel, piano key C object tag assignment. Available piano key object tags listed

Further discussing the relationship between ray casting and player interaction, alongside touch-based interaction, ray casting was also adapted for look-based interaction with objects; this requires the player to align an interactable object in the centre of the first-person cameras field-of-view (FOV), when being within close proximity of it. Said technique was particularly useful for simulating motion-sensed lighting, which was used for identifying object interactions, sequentially. Moreover, look-based interactions were also implemented for entering object interaction views, and as an initiative for presenting narrative development; this can be seen within the second puzzle room.

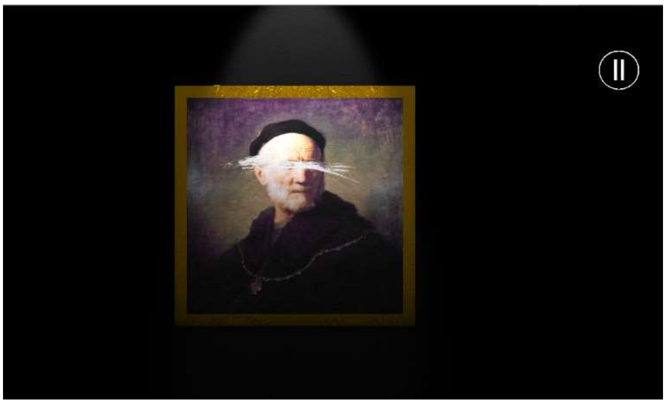


*Figure 18: Puzzle room two, music box picture, narrative development scene. Music box picture animates upon the player being close by and looking at the picture canvas directly*

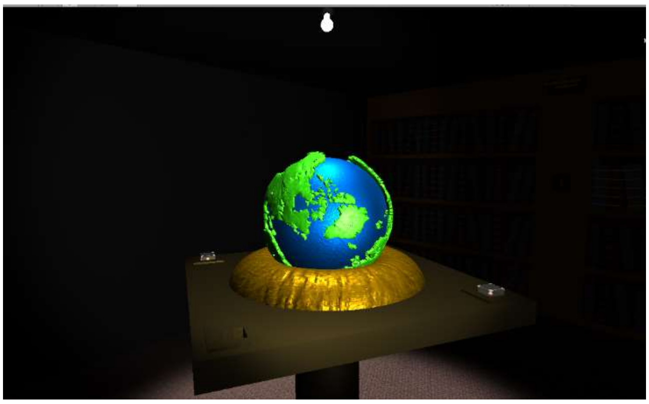


*Figure 17: Puzzle room two, table piece, hanging light animates and brightens the area of the scene, upon the player being close by and looking at the table globe directly*

In addition to touch and look-based interaction via ray casting, through the use of peripheral and interface input, a player is able to interact with objects by performing swipe, drag, pinch and pinch rotate gestures; all of the which are supported by the game to visualise the manipulation of objects, similar to the way players would anticipate real-world object interactions and other existing mobile puzzle game interactions [2] [5] [6] [13] [14]. As outlined in the functional requirements for the game, player-to-object interaction should consider hold and swipe gestures, supplementary to touch; this aims to familiarise players with previously played game mechanics, for avoiding early complication and frustration. Meanwhile, as found within the literature review, puzzle games intend to provide players with interesting interactions that accumulate challenge; object interactions that are exclusively touch and swipe dependant, cannot accompany this ambition given their simplicity. However, as pinch, and rotatory gestures are supported as object interaction techniques, these production values are fulfilled, and hence were necessary.

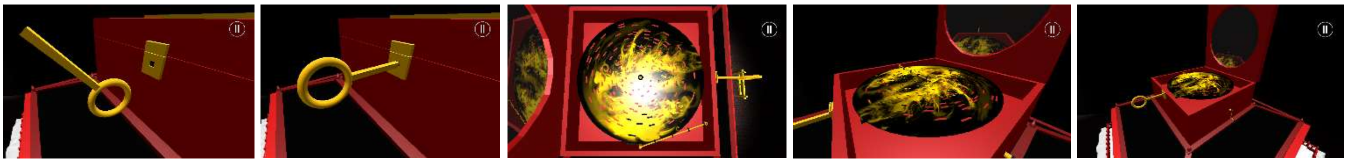


*Figure 19: Puzzle room two, music box puzzle interaction. Music box latch key rotate, latch key position translation, disc spindle rotate, camera position translation inwards, camera position translation outwards; touch and drag, swipe right, pinch, and rotate left, pinch inwards, pinch outwards. Captioned left to right*

## Camera Application

For adhering to the first-person reception of the game, the application of the first-person camera had greater significance aside from providing a visualisation of the game world[18]. Functionally, the facing direction of the first-person camera performs as the player objects direction of forward and backward modes of traversal[19]. As well, the first-person camera is used to characterize the animations of the player object, to reduce the threshold for error within collision detection, camera position translation, and up rise in performance overhead, as opposed to the player object being animated directly. However, when the first-person camera is disabled, the player cannot interact with objects, and in result, the player cannot progress through the game; as mentioned in sections prior, forcing the first-person perspective aims to heighten the realism, believability, and horror

[18] From the playable characters perspective.

[19] This is described within the functional requirements document (see ***Appendix B***).

attributes of the game. This technique has wider implications on the theme of suspense envisioned for the game[20]; where the use of the first-person perspective targets player immersion, from sharing the same visual and audible information as the player-controlled character. This was noticed from my interaction and study upon the computer game title, Slender: The Eight Pages [20]. As documented within the game's functional requirements, object interaction via the first-person perspective also forms as part of the software's use case scenarios; it was thereby essential that this mechanism was addressed.

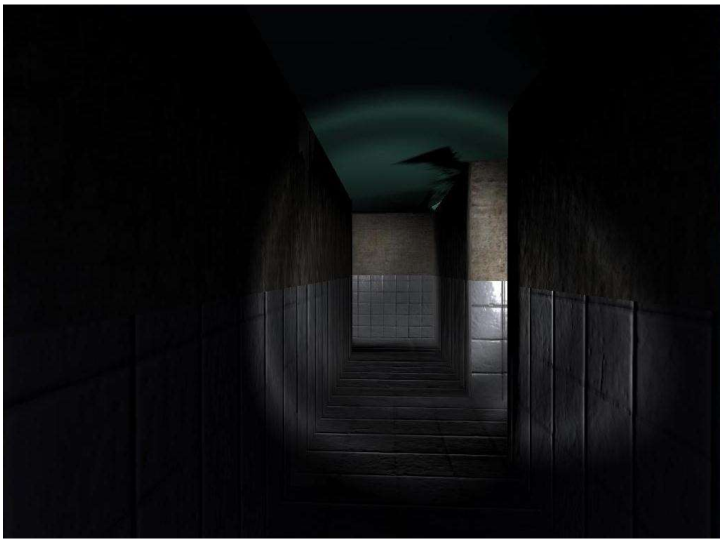


*Figure 20: Slender: The Eight Pages, first-person perspective of an indoor corridor environment [21]*

Mandatory to object interaction and narrative development, the first-person cameras primary function is to captivate visual information, that prompts players with signs that steer the progression and narrative plot of the game; such information is issued within object interaction views, animation sequences and audio cues throughout the game. As often referred to throughout the documentation of this project, this adaptation of the first-person camera perspective was put forth by the studies of The Room [2] and The House of Da Vinci [7] mobile games. Using this mechanism of the first-person camera has enabled the pace of gameplay to be slowed, as intended, whilst allowing players to be challenged, yet superficially navigated though the game.

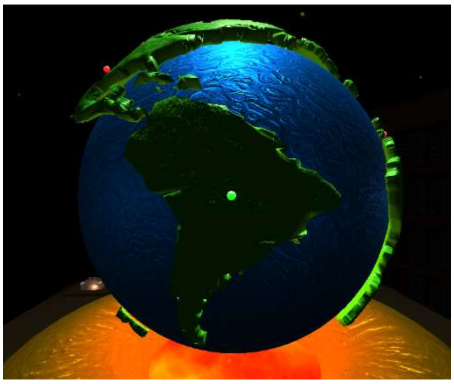


*Figure 23: Puzzle room two, table piece puzzle, table globe object interaction view, table piece puzzle iteration one animation*

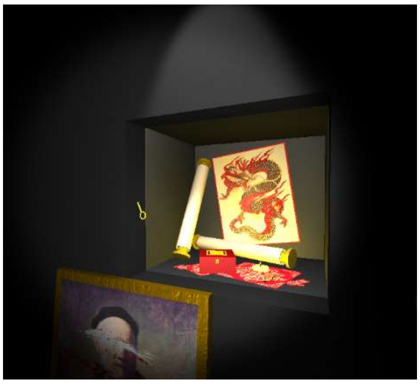
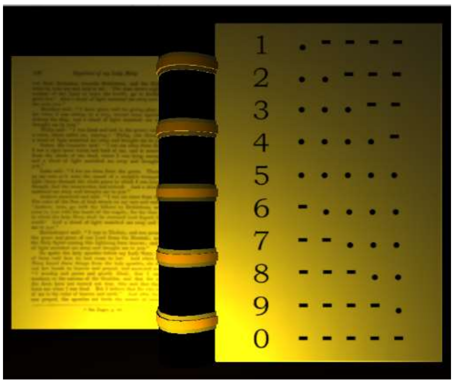


*Figure 22: Puzzle room two, music box puzzle, music box latch key object animation*



1 .----
2 ..---
3 ...--
4 ....-
5 .....
6 -....
7 --...
8 ---..
9 ----.
0 -----

*Figure 21: Puzzle room one, telephone puzzle, audio device Morse code translation, Morse code book object interaction view*

[20] As mentioned within the literature review (see ***Appendix A***).

Without the first-person cameras ability to translate between object interaction views and the player object, the game would seem overly complex from an implementation standpoint. Moreover, the game would appear unconventional and dysfunctional to the genre and provide fewer interesting interactions, that would also be impractical for the precision and scale of a mobile interface[21].

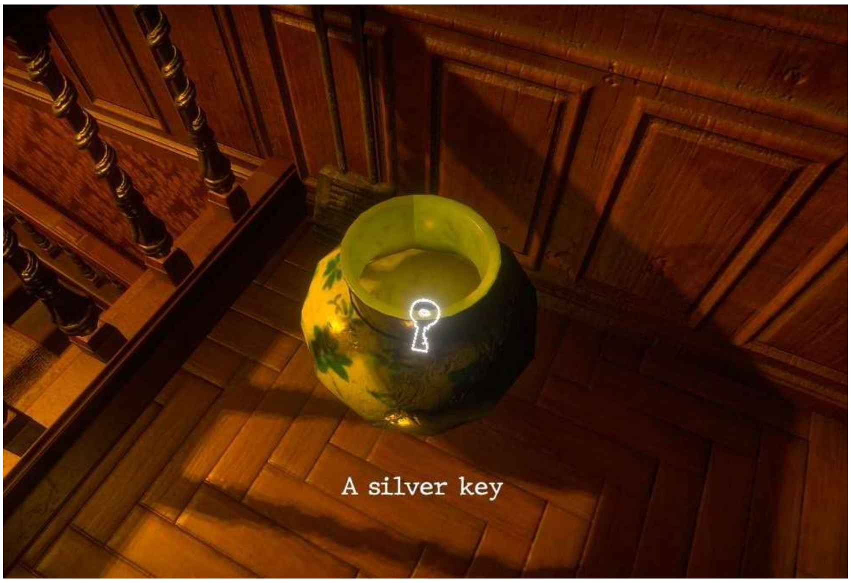


*Figure 24: Mad Experiments: Escape Room, first-person camera perspective fixated to the player-controlled character, roam-focused interaction [23]*

## User Interface

Fundamental to mobile games, UI's are particularly useful for addressing uncomplicated interactions and being able to access and manage game settings, that administer select features of the application. In accordance with the functional requirements of the game, UI's pose significance to a players ability to invoke the movement and interaction functionality of the player-controlled object; in addition to initiating, pausing, resuming and terminating the current game session, and toggling subtitles that are visualised as overlaying text.

Controlling player movement and invoking object interaction via UI elements is typically unseen across existing mobile puzzle games, which alternatively exhaust the use of gesture interactions to achieve. However, as seen within The Room [1] and The House of Da Vinci [5] mobile games, it is understood that the instance of the camera does not imitate a physical presence and operates dissimilarly to a player-controlled object; this design choice contradicted my intentions of engaging player interest and suspense through realism, and so the Call of Duty: Mobile [9] and PUBG: Mobile [10] styles of user interface were adapted for this purpose. Being an unconventional characteristic of the genre, does not hinder the production values of the application; instead, it enables the game to differentiate from the genre for exclusivity purposes, alongside forming an immersive experience for players[22].

Aside from player interaction methods aforementioned, player-related control was appropriated by the presence of a multidirectional-joystick and series of interactable buttons; these design choices were necessary for deviating from the techniques employed by object interaction, addressing the

[21] Interactions of this type are typically roaming-focussed, as can be seen within Mad Experiments: Escape Room [22].

[22] This was mentioned within the literature review (see ***Appendix A***).

dynamic of game interactions, and for adhering to the control of a player character, as opposed to a camera object. The requirement for controlling a player object is mandatory for inciting game realism[23]. For the application of a multidirectional-joystick, it was preferred in comparison to a series of directional buttons, that would each control a single mode of traversal; given its compatibility with continuous and intermediated player traversal [24].

Meanwhile, in relation to the visibility of the UI, being able to visualise the UI elements was essential for invoking the player-controlled objects upward mode of traversal, toggling flashlight activeness and toggling the active camera perspective. If said mechanics of the game were not impactful to the style of gameplay presented, a series of touch fields could have been incorporated alternatively, as seen within the mobile puzzle game, realMyst [25]; using touch fields as opposed to visible buttons would have benefit the cinematography ideals of the game, which would yet again be a focus of heightening the games realism and suspense outlooks.

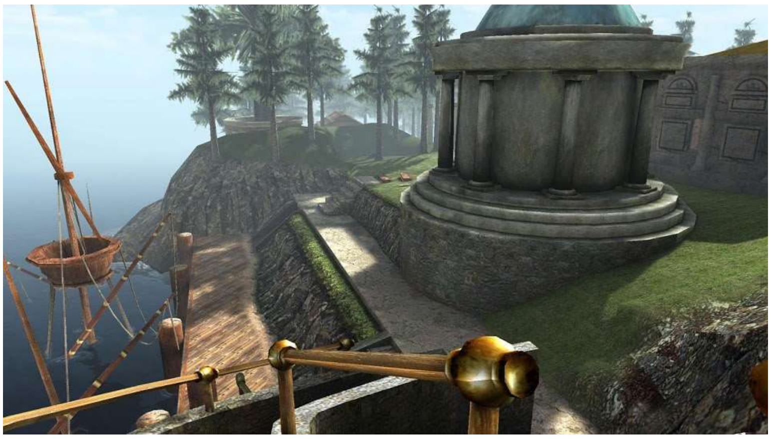


*Figure 25: realMyst, game world cinematograph captivation, UI elements not visible to player, touch fields used to control player traversal through the world [26]*

As for the administration of game settings, the game required an additional user interface to address the user case scenarios relevant to subtitle toggling and for determining the functional state of the application. Similar to the physical design of the user interface prior, a series of buttons exist to address the functional requirements of the pause menu; this design choice optimises the games simplicity and puts forth a normalised procedure for managing application states[24].

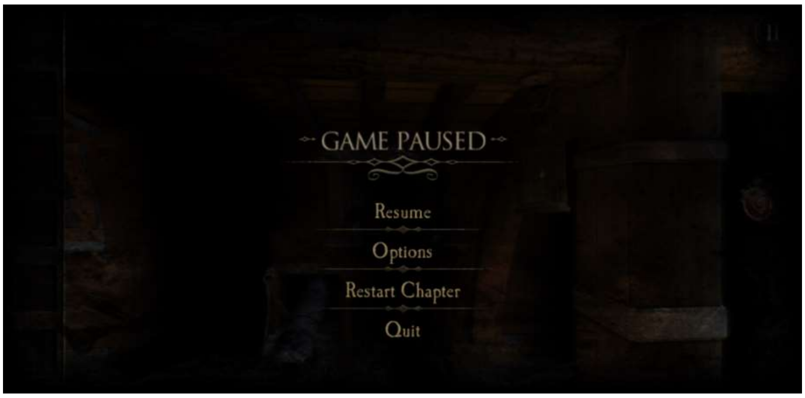


*Figure 26: The Room Two, game paused state, pause menu interface (captured from the game directly)*

[23] This is outlined within the functional requirements document (see ***Appendix B***) and referred to within sections prior.

[24] A mobile puzzle game that best illustrates this, is The Room Two [27].

# System Design

## System Architecture

In accordance with the systems design objectives[25], the deliverable state of the game was expected to address camera object control, player object control, player interaction, puzzle sequencing, navigable configuration menus and interfaces, as well as audio cue subtitling. For adhering to the implementation of these functional requirements, the preliminary Unified Modelling Language (UML) diagram featured in the SDD was referred to, as the basis for the games back-end architecture implementation, during the initial development stages of the software.

Relating to the architecture proposed in the preliminary UML diagram, it was anticipated that the functional requirements of the game were componentised into classes of separate functionality, which when combined, resemble a model-view-controller (MVC) design pattern; this was intentionally designed for isolating the functional requirements of the game, maintaining the robustness of the applications code base and issuing application updates to player input, through event handling [28].

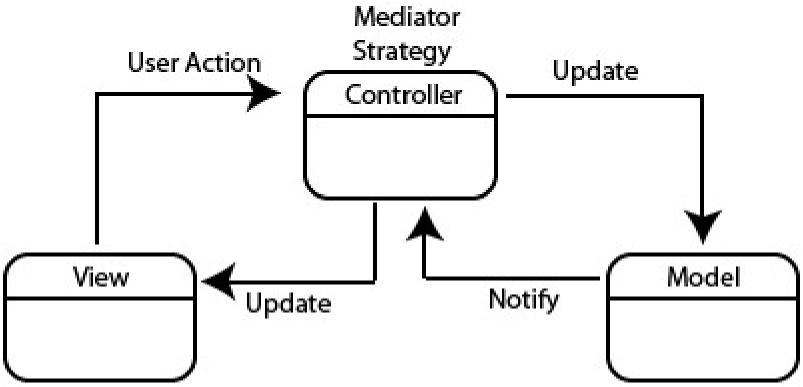


*Figure 27: Model-view-controller (MVC) design pattern [29]*

In continuation of design patterns, the preliminary architecture of the application also considered the use of object-orientated programming (OOP) principles[26], for maintaining the organisation of the applications code base, and optimising the applications pace of development[27]. Whereas functionally, the inclusion of the OOP principles were purposed for addressing similar functionality between classes and preventing unintentional modifications to another class's members.

Since the initial design and development of the application, the back-end architecture of the game has been modified drastically from the preliminary intent, as a result of unforeseen error, extension to the games interaction proposal and the underestimation of method density. A noticeable difference between the final and preliminary designs is the absence of the 'AudioController' class, which throughout the development of the application, had no significance to the robustness of the applications code base, and was instead a hindrance to the simplicity of invoking audio cues[28]. To overcome this impediment, audio sources were retargeted as components on game objects being

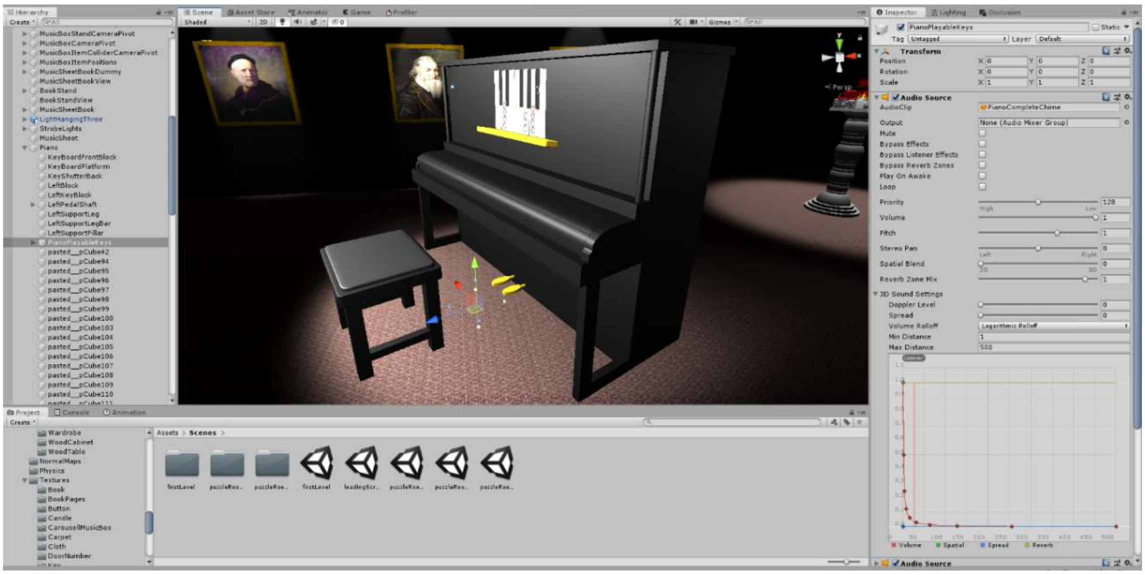
[25] Specified within the system design document (SDD) (see ***Appendix D***).

[26] OPP principles considered were abstraction, encapsulation, inheritance, and polymorphism.

[27] As intended and referred to within the system design document (SDD) (see ***Appendix D***).

[28] Where multiple definitions of identical game objects existed in discrete classes.

accessed by classes referencing them already; as opposed to their existing game objects for addressing audio, exclusively.



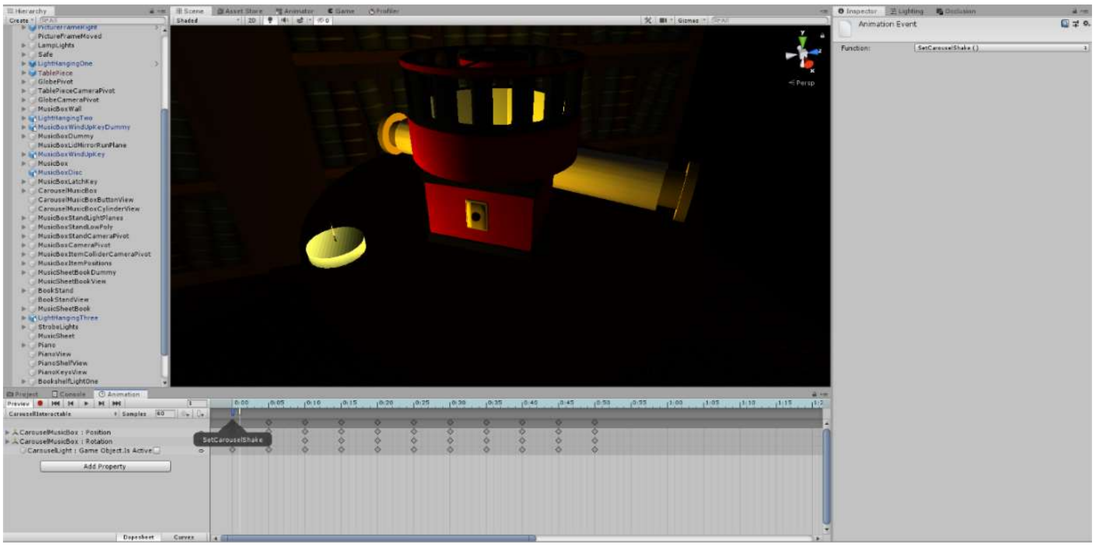
PianoPlayableKeys
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Audio Source
AudioClip PianoCompleteChime
Output None (Audio Mixer Group)
Mute ☐
Bypass Effects ☐
Bypass Listener Effects ☐
Bypass Reverb Zones ☐
Play On Awake ☐
Loop ☐
Priority 128
Volume 1
Pitch 1
Stereo Pan 0
Spatial Blend 0
Reverb Zone Mix 1
3D Sound Settings
Doppler Level 0
Spread 0
Volume Rolloff Logarithmic Rolloff
Min Distance 1
Max Distance 500

*Figure 28: Puzzle room two, 'PianoPlayableKeys' object, numerous audio sources seen attached as components within Unity's inspector panel*

Additional to the back-end architecture envisioned in the preliminary design, many classes reserved for object animation have also been incorporated[29]. Their existence within the application was necessary for utilizing animation events[30], which enable the narrative development of the game to be driven and depicted. However, as a restraint of the Unity engine [30], for the use of animation events, it is required that the methods that animation events invoke are separated by class and script; as opposed to existing collectively, which is preferred[31].

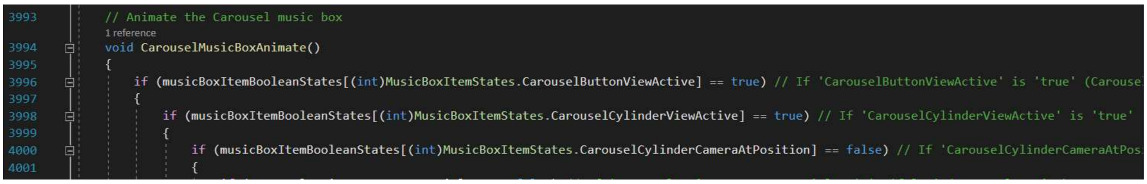


Animation Event
Function: SetCarouselShake ()

*Figure 29: Puzzle room two, carousel music box object, animation timeline within the Unity engine, showcasing the presence of animation events. Method being invoked by the animation event is shown within the inspector panel, and within the animation timeline as a widget*

[29] These classes present associative relations to the game's fundamental classes.
[30] These are used to conditionally invoke other functionality during object animation sequences.
[31] This informed my decision for creating classes independently of each animated object.

Moreover, accounting for the readability of the applications code base, enumeration was adapted for restructuring the way in which class members were indexed. Offering a linguistic representation for indexing class members was insignificant to the performance of existing functionality, however, its application was necessary for addressing syntactical intricacies of class members, that rose in quantity over time[32].



```
// Animate the Carousel music box
1 reference
void CarouselMusicBoxAnimate()
{
    if (musicBoxItemBooleanStates[(int)MusicBoxItemStates.CarouselButtonViewActive] == true) // If 'CarouselButtonViewActive' is 'true' (Carouse
    {
        if (musicBoxItemBooleanStates[(int)MusicBoxItemStates.CarouselCylinderViewActive] == true) // If 'CarouselCylinderViewActive' is 'true'
        {
            if (musicBoxItemBooleanStates[(int)MusicBoxItemStates.CarouselCylinderCameraAtPosition] == false) // If 'CarouselCylinderCameraAtPos
            {
```

*Figure 30: 'PuzzleRoomTwo' class, conditional use of enumeration for indexing Boolean array members*

In further relation to class members and the preliminary architecture of the application, the use of container class members were not forecast in the initial UML diagram produced, whereas the final architecture displays coherent use of lists and arrays[33]. Throughout the development of the applications code base, it was acknowledged that the functional density of class methods required more variables to address the behaviours intended, than what was projected in the initial design. In result, the games development slowed due to the depreciation of code organisation, that was inhabited from an increase in variable declaration and initialisation statements. The insertion of arrays and lists better presented the application code when combined with the enumeration mentioned prior, which inclined the applications robustness and pace of development[34].

## User Interfaces

Relating to the preliminary UI expectations of the application[35], multiple interfaces were necessary for delivering player-controlled character movement and interaction, alongside initiating, pausing, resuming, and terminating the current game session, and toggling the activeness of the integrated subtitle system[36].

In the order of implementation, the initial design for the loading screen UI, functionally expected the initiation and termination of the game session, alongside being able to toggle the active state of the subtitle system. In accordance with the functional requirements document, it was projected for said functionality to be invoked via the application of interactable buttons; which were aesthetically designed to resemble an ascending arrangement, with intent of being unconventional to existing menu designs.

With regards to the applications adherence to the design put forth, the UI implements all of the functional requirements mentioned, for fulfilling the games use case scenarios from the perspective of a player; this has been achieved through the support of interactable buttons, which physically

[32] This was a defect of the applications code base expanding over the course of its development and was not anticipated within the preliminary design.

[33] These variable types have presence in the majority of the systems classes.

[34] There are also reduced implications to performance, which was a design constraint considered within the system design document (see ***Appendix D***); for the modernized, back-end architecture of the game, see ***Appendix H***.

[35] Referred to in the functional requirements for the game (see ***Appendix B***),

[36] All of the functionality implemented and referred to within the interfaces, are governed by the 'WindowGUI', 'SceneController' and 'ButtonController' classes (see ***Appendix H***). The user interface implementations present the front-end architecture of the application.

conform to the initial design layout. Meanwhile, separate to the interface's behaviours, the game title field[37] has been substituted for a static image, representing the games brand; this offers an enhancement to the scene's aesthetic, as opposed to using text. The images position within the UI canvas also differentiates from the game title fields; this aims to captivate the cinematography of the scene, effectively.

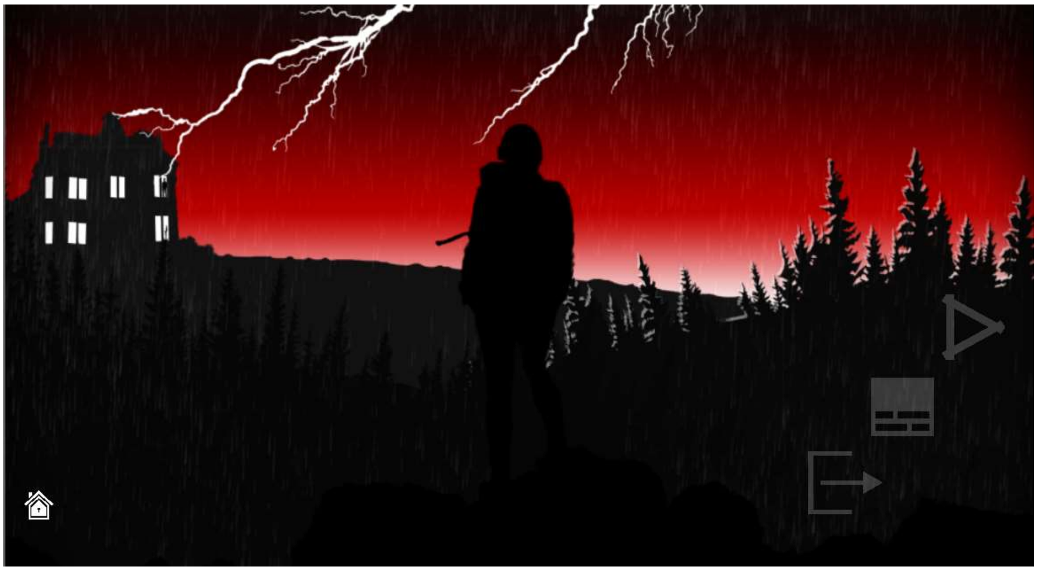


*Figure 31: Loading screen scene, showcasing the loading screen menu user interface design*

Relating to the initial design of the pause menu UI[38], players were expected to resume and exit the current game session, in addition to toggling the subtitle systems active state. The invocation of said functionality was also proposed by a composition of interactable buttons[39], which aimed to be anatomically similar to a list. This purposed to prevent unintentional interactions with other buttons in the interface.

As can be observed from the pause menu design delivered, the UI adheres to its functional requirements through the adaptation of interactable buttons; each representing and administering the resume, subtitle toggle and exit features of the game. Contesting the preliminary design layout of the pause menu UI, the final design of the UI overlooks the menu title field, for adhering to the demand for simplicity, and non-linguistic representations of game elements. Upon the preliminary UI design being delivered, the implications of presenting a menu title were not yet realised for addressing the multi-cultural learning criterion of the game[40].

[37] Detailed in the functional requirements document (see ***Appendix B***).

[38] Aligned with the pause menus functional requirements (see ***Appendix B***).

[39] As seen within the loading screen user interface.

[40] This is detailed within the global checklist (see ***Appendix F***).

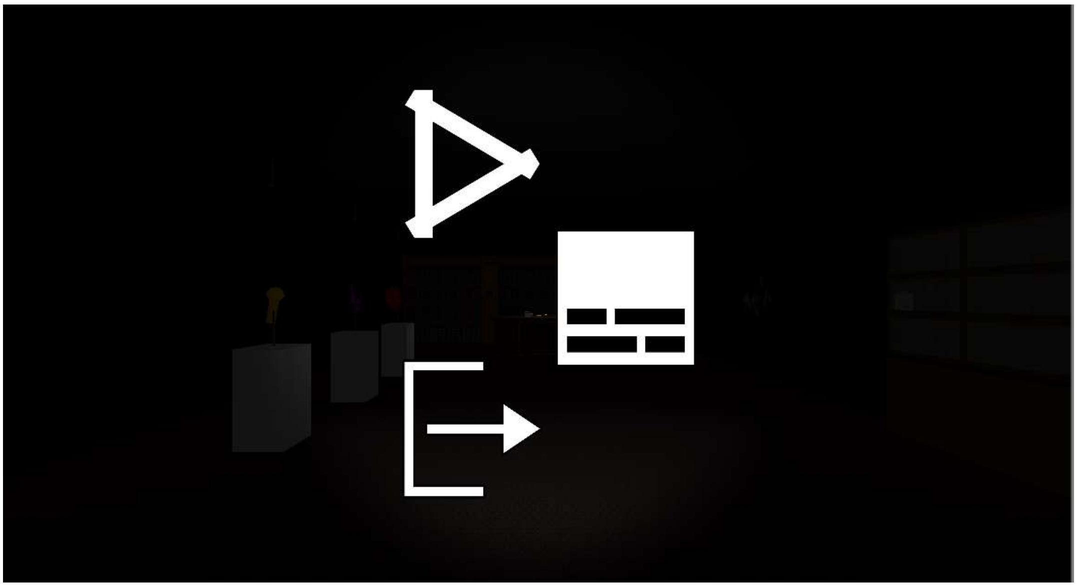


*Figure 32: Pause menu user interface, displaying the resume, toggle subtitle and exit application buttons*

Lastly, with regards to the preliminary design of the in-game UI, it was significant that the interface addressed the players ability to invoke player-controlled character traversal, to toggle the active state of the flashlight, to alternate between active camera perspective and to invoke the presence of the pause menu interface; all through the inhabitation of interactable buttons[41]. Furthermore, the initial interface design also details an image field, purposed for visualising items that the player obtains; this was expected to be physically dependent of the UI's functional elements[42]. For which, the partitioning of UI elements portrayed within the interfaces preliminary design, was targeted to evading viewport obstruction, addressing the landscape orientation of gameplay, and for preventing unintentional functional invocation[43].

As noticed from the finalised in-game UI design, the image fields that intended to exhibit obtained items were not implemented; this was due to their insignificance in enabling players to identify gameplay objectives and for their contribution to cluttering camera viewports. Alternatively, the principles for the image fields were addressed by regulating similarities in object geometry and their material properties. Relating to player-controlled character traversal, interactable buttons were subsequently replaced for a single multidirectional-joystick, for the support of continuous and intermediate adjustments to the player objects movement speed, and supplementary audio cue dynamics[44].

Additionally, the physical arrangement of the camera toggle and jump buttons discriminate from their alignments projected within the preliminary design; this adaptation clusters the functional invocation of the interface, into separate physical areas for its ease-of-use. Aside from the aesthetic discrepancies between the UI designs, all of the functional requirements expected from the interface, are addressed; which fulfils the front-end user requirements for the application.

[41] This is referred to within the functional requirements document (see ***Appendix B***) and is reprocessed from the user interfaces discussed prior.
[42] This is not obligatory to the applications functionality, as mentioned within the functional requirements document (see ***Appendix B***).
[43] This is a design concept that is shared with the pause menu user interface.
[44] As mentioned in the ***Background*** section.

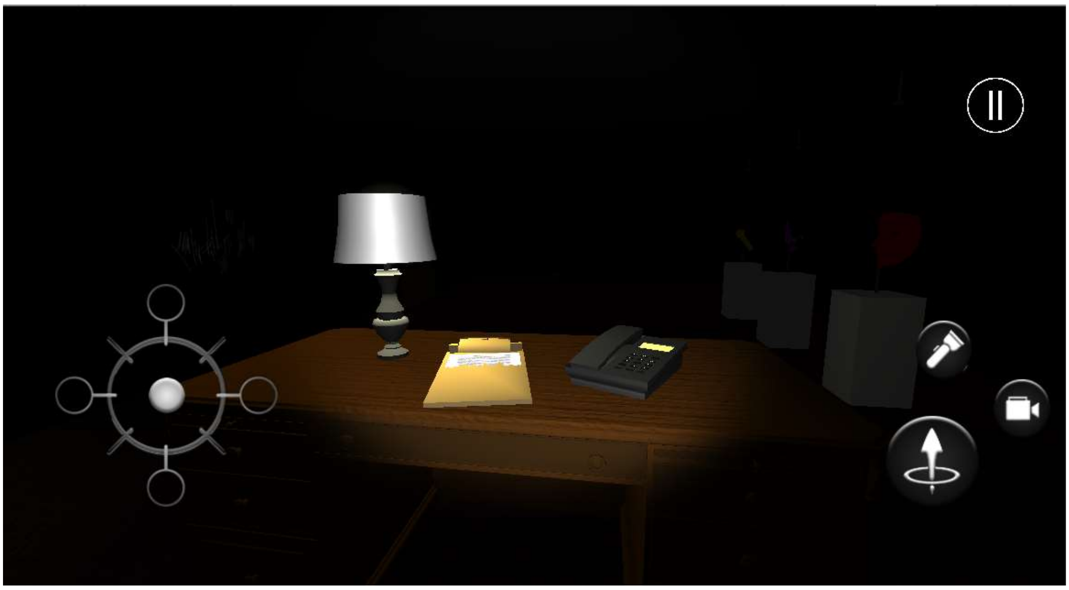


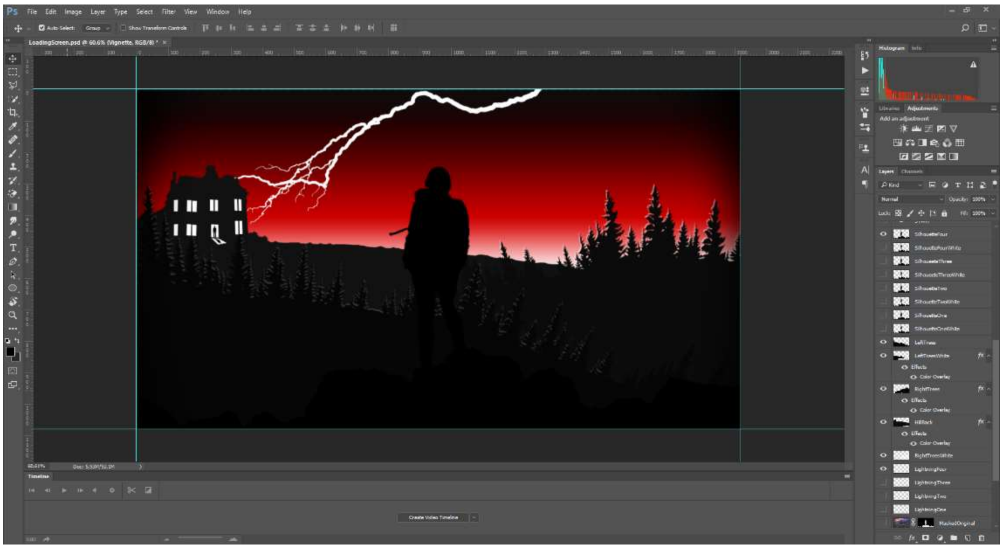
*Figure 33: In-game user interface, showcasing the joystick, jump, flashlight toggle, camera perspective alternate and pause menu buttons*

# System Implementation

## Software Development

### Loading Screen

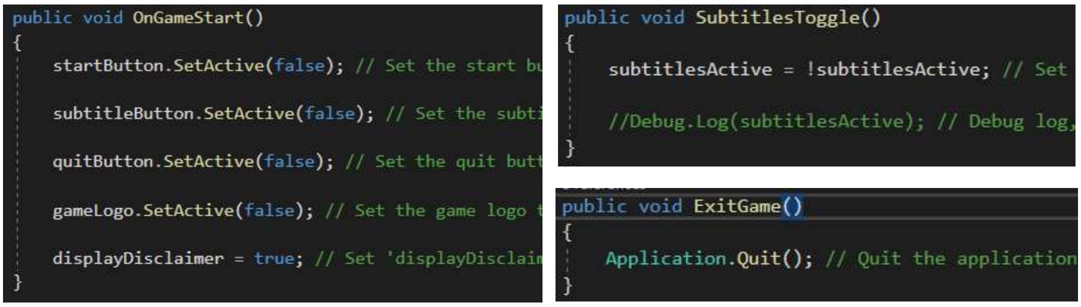
For the initial development process of the loading-screen scene, a series of graphical images were rendered with the software application, Adobe Photoshop [36], which prepared the front-end architecture of the scene[45]; by addressing the desired aesthetic[46], animated components and alignment for the UI elements[47].



[45] Images included the backgrounding imagery for the scene, button textures and the game logo.
[46] Referred to in the ***Background*** section.
[47] This is mentioned within the system design document (see ***Appendix D***), and in the section prior.

*Figure 34: Adobe Photoshop CC 2017, displaying the software's interface and the UI elements created for depicting the loading screen*

Beyond the scene's aesthetic development, a series of interactable buttons were placed within the UI canvas to address the scenes functional requirements; to initiate and exit a game session, and to toggle the active state of the subtitle system[48], as a player. This functionality was adhered to in the initial development of the 'WindowGUI', 'SceneController', 'ButtonController' and 'LoadingScreenAnimator' scripts[49]; where the start, toggle subtitle and exit application buttons invoke the 'OnGameStart'[50], 'SubtitlesToggle'[51] and 'ExitGame'[52] methods[53].



```
public void OnGameStart()
{
    startButton.SetActive(false); // Set the start bu

    subtitleButton.SetActive(false); // Set the subti

    quitButton.SetActive(false); // Set the quit butt

    gameLogo.SetActive(false); // Set the game logo t

    displayDisclaimer = true; // Set 'displayDisclai
}
```

```
public void SubtitlesToggle()
{
    subtitlesActive = !subtitlesActive; // Set

    //Debug.Log(subtitlesActive); // Debug log,
}
```

```
public void ExitGame()
{
    Application.Quit(); // Quit the application
}
```

*Figure 35: 'WindowGUI', 'ButtonController' and 'SceneController' classes, representing the functionality of the methods directly invoked by the interactable buttons*

Additional to this functionality, a graphical user interface (GUI) resembling a disclaimer window was implemented, as the functional invocation for the start button. This was a mandatory design feature for informing players about the games theme of content[54]; players are able to acknowledge the disclaimer via interacting with it, as delivered in the 'OnGUI' and 'DisplayDiscliamerGUI' methods[55].

[48] This is referred to in previous section of this document.

[49] ***Appendix H***.

[50] Within the 'WindowGUI' class (see ***Appendix H***).

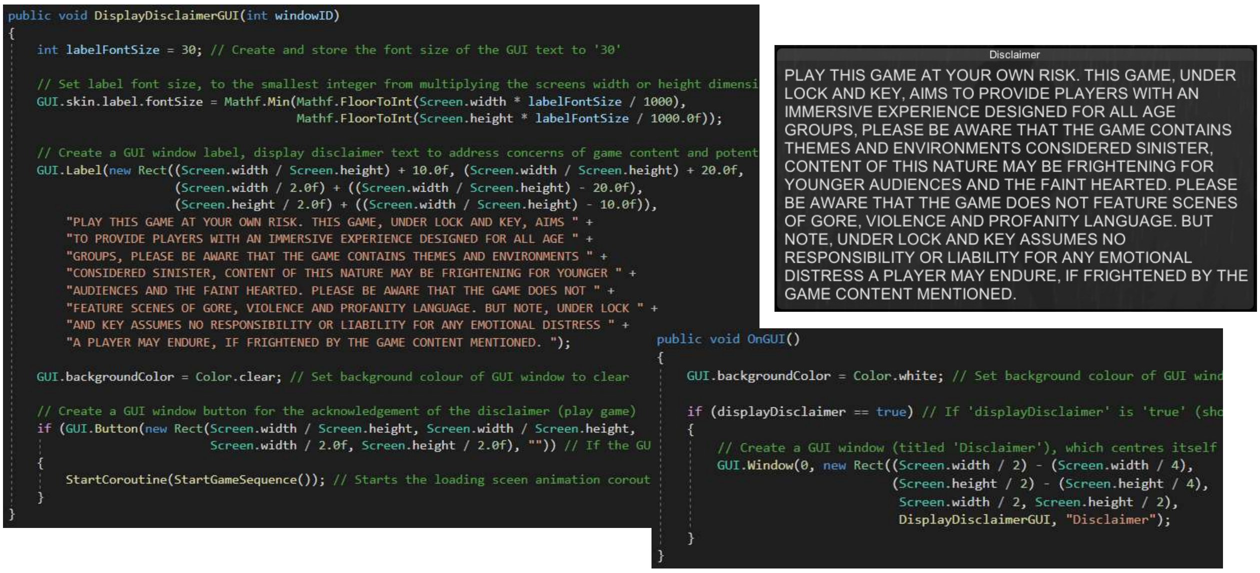
[51] Within the 'ButtonController' class (see ***Appendix H***).

[52] Within the 'SceneController' class (see ***Appendix H***).

[53] ***Appendix H.***

[54] Referred to within the ethical review document (see ***Appendix G***).

[55] Within the 'WindowGUI' class (see ***Appendix H***).



```
public void DisplayDisclaimerGUI(int windowID)
{
    int labelFontSize = 30; // Create and store the font size of the GUI text to '30'

    // Set label font size, to the smallest integer from multiplying the screens width or height dimensi
    GUI.skin.label.fontSize = Mathf.Min(Mathf.FloorToInt(Screen.width * labelFontSize / 1000),
                                        Mathf.FloorToInt(Screen.height * labelFontSize / 1000.0f));

    // Create a GUI window label, display disclaimer text to address concerns of game content and potent
    GUI.Label(new Rect((Screen.width / Screen.height) + 10.0f, (Screen.width / Screen.height) + 20.0f,
                       (Screen.width / 2.0f) + ((Screen.width / Screen.height) - 20.0f),
                       (Screen.height / 2.0f) + ((Screen.width / Screen.height) - 10.0f)),
        "PLAY THIS GAME AT YOUR OWN RISK. THIS GAME, UNDER LOCK AND KEY, AIMS " +
        "TO PROVIDE PLAYERS WITH AN IMMERSIVE EXPERIENCE DESIGNED FOR ALL AGE " +
        "GROUPS, PLEASE BE AWARE THAT THE GAME CONTAINS THEMES AND ENVIRONMENTS " +
        "CONSIDERED SINISTER, CONTENT OF THIS NATURE MAY BE FRIGHTENING FOR YOUNGER " +
        "AUDIENCES AND THE FAINT HEARTED. PLEASE BE AWARE THAT THE GAME DOES NOT " +
        "FEATURE SCENES OF GORE, VIOLENCE AND PROFANITY LANGUAGE. BUT NOTE, UNDER LOCK " +
        "AND KEY ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY EMOTIONAL DISTRESS " +
        "A PLAYER MAY ENDURE, IF FRIGHTENED BY THE GAME CONTENT MENTIONED. ");

    GUI.backgroundColor = Color.clear; // Set background colour of GUI window to clear

    // Create a GUI window button for the acknowledgement of the disclaimer (play game)
    if (GUI.Button(new Rect(Screen.width / Screen.height, Screen.width / Screen.height,
                            Screen.width / 2.0f, Screen.height / 2.0f), "")) // If the GU
    {
        StartCoroutine(StartGameSequence()); // Starts the loading sceen animation corout
    }
}
```

Disclaimer

PLAY THIS GAME AT YOUR OWN RISK. THIS GAME, UNDER LOCK AND KEY, AIMS TO PROVIDE PLAYERS WITH AN IMMERSIVE EXPERIENCE DESIGNED FOR ALL AGE GROUPS, PLEASE BE AWARE THAT THE GAME CONTAINS THEMES AND ENVIRONMENTS CONSIDERED SINISTER, CONTENT OF THIS NATURE MAY BE FRIGHTENING FOR YOUNGER AUDIENCES AND THE FAINT HEARTED. PLEASE BE AWARE THAT THE GAME DOES NOT FEATURE SCENES OF GORE, VIOLENCE AND PROFANITY LANGUAGE. BUT NOTE, UNDER LOCK AND KEY ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY EMOTIONAL DISTRESS A PLAYER MAY ENDURE, IF FRIGHTENED BY THE GAME CONTENT MENTIONED.

```
public void OnGUI()
{
    GUI.backgroundColor = Color.white; // Set background colour of GUI wind

    if (displayDisclaimer == true) // If 'displayDisclaimer' is 'true' (sho
    {
        // Create a GUI window (titled 'Disclaimer'), which centres itself
        GUI.Window(0, new Rect((Screen.width / 2) - (Screen.width / 4),
                               (Screen.height / 2) - (Screen.height / 4),
                                Screen.width / 2, Screen.height / 2),
                                DisplayDisclaimerGUI, "Disclaimer");
    }
}
```

*Figure 36: 'WindowGUI' class, displaying the methods invoked by the disclaimer window. The disclaimer window GUI is shown in top-right region of figure*

Advancing from the disclaimer prompt, a series of animations were applied to the UI images created prior; this was achieved using Unity's animator interface and animation timeline. Said animations were implemented for the enhancement of the scene's cinematography[56] and were invoked sequentially, via the 'StartGameSequence' enumerator method[57].

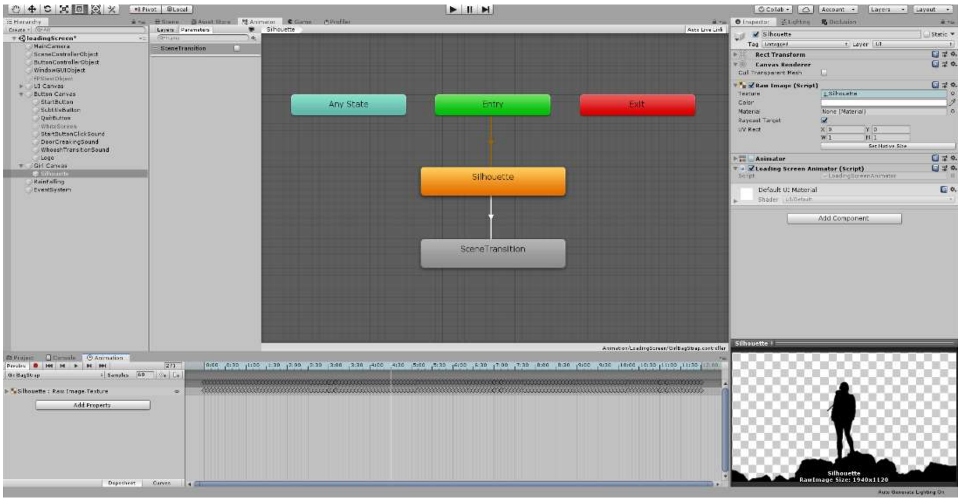


*Figure 37: Unity's animator interface, showcasing the relations between the silhouette objects animation states. Animation timeline is seen in the bottom region of the applications interface*

## Base Scene

Regarding the initial development procedures for the base scene, an array of light[58] and three-dimensional objects[59] were populated and merged by hierarchy within Unity's editor interface; this purposed for visualizing the hub to the room-like environments, proposed for the games escape-the-

[56] With the intent of captivating player interest prematurely, as discussed in the ***Background*** section.

[57] Within the 'WindowGUI' class (see ***Appendix H***), this method is invoked upon the disclaimer window being acknowledged.

[58] Used to highlight room numbers and to increase the visibility of the scene.

[59] Cubes, planes, and particle systems were used.

room orientation[60]. For which, three-dimensional modelling applications were not approached[61], due to the low-polygonal designs of objects[62], that could be addressed within Unity[63].

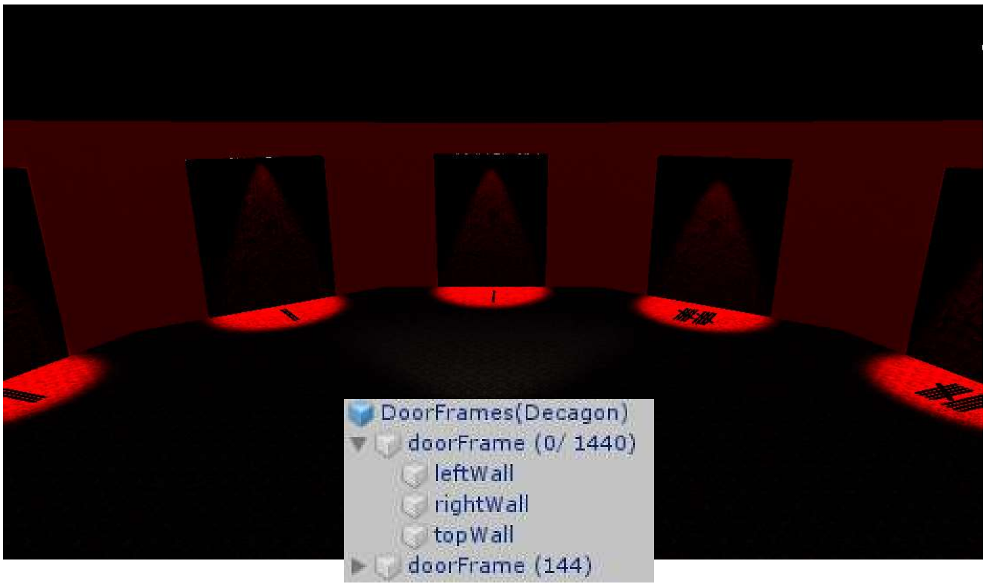
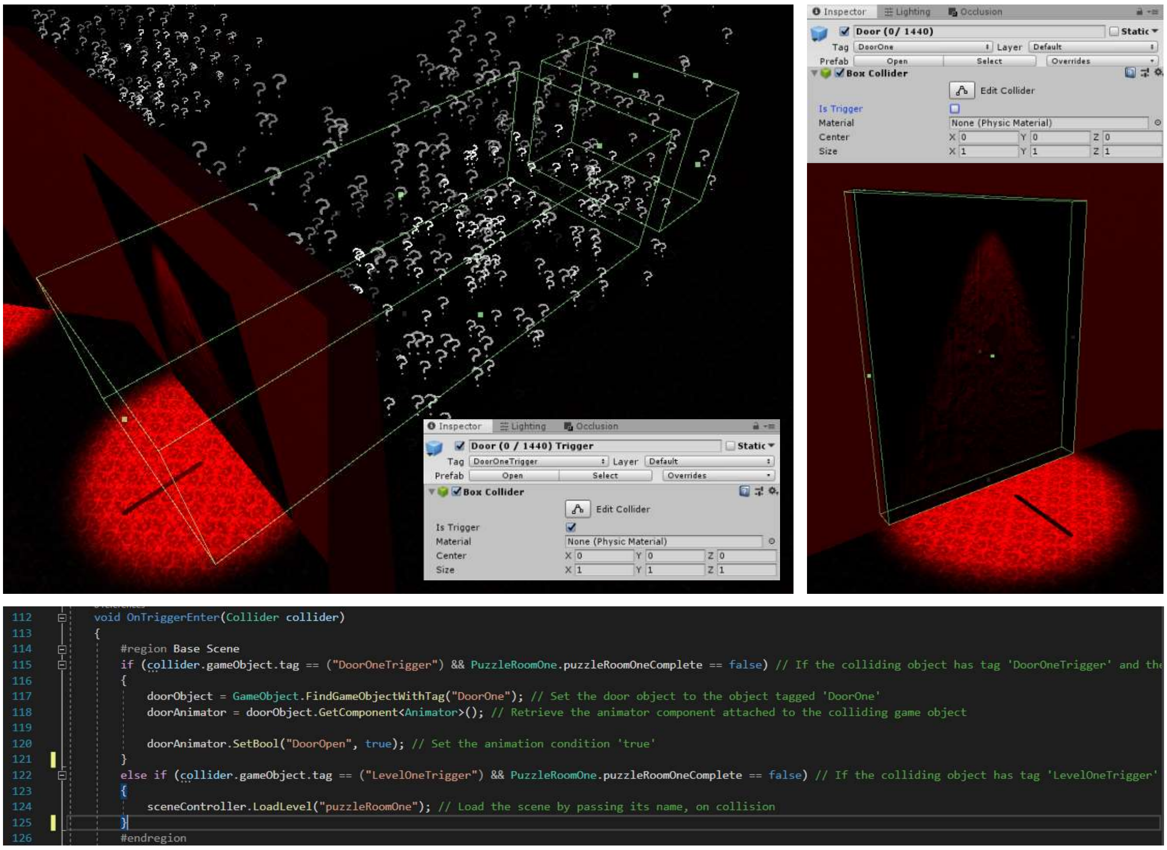


*Figure 38: First level scene, displaying the decagonal arrangement of door and wall objects. Game object hierarchy is also demonstrated*

Alongside the aesthetic application of the scene, a series of empty game objects were also populated in the scene, for addressing the player-controlled characters requirement to exist in multiple environments[64]. To achieve such, all but two of the objects[65] were attached with a collider component, configured as a trigger[66]; this was necessary for checking collisions that were registered by the player object[67]. For authoring the third and first-person perspectives of the game[68], the player object was used to hierarchically parent two camera objects[69].

[60] Discussed within the ***Background*** section.
[61] Such as Autodesk Maya, previously referred to.
[62] Using three-dimensional modelling applications is more time consuming for unsophisticated models.
[63] Unity engine was sufficient and more efficient for modelling unsophisticated objects.
[64] For demonstrating the interconnectivity of the puzzle room scenes, and for fulfilling player-character control. This is mentioned in the ***Background*** section,
[65] Reserved for the player-controlled character, and its flashlight child object.
[66] The player-controlled character applied a capsule collider, not configured as a trigger, for the purpose of collision detection capabilities.
[67] For the purpose of invoking door object animations, and loading other game scenes appropriately, with consideration to the players progression. This is addressed by the 'OnTriggerEnter' method, within the 'PlayerController' class (see ***Appendix H***).
[68] Detailed as a functional requirement for the game (see ***Appendix B***).
[69] Each camera object was configured with a different field-of-view (FOV) and local-position, relative to the player object. This enabled a perspective difference to be stimulated.



Door (0 / 1440) Trigger □ Static
Tag DoorOneTrigger Layer Default
Box Collider
Is Trigger ☑
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

Door (0/ 1440) □ Static
Tag DoorOne Layer Default
Box Collider
Is Trigger □
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

```
void OnTriggerEnter(Collider collider)
{
    #region Base Scene
    if (collider.gameObject.tag == ("DoorOneTrigger") && PuzzleRoomOne.puzzleRoomOneComplete == false) // If the colliding object has tag 'DoorOneTrigger' and the
    {
        doorObject = GameObject.FindGameObjectWithTag("DoorOne"); // Set the door object to the object tagged 'DoorOne'
        doorAnimator = doorObject.GetComponent<Animator>(); // Retrieve the animator component attached to the colliding game object

        doorAnimator.SetBool("DoorOpen", true); // Set the animation condition 'true'
    }
    else if (collider.gameObject.tag == ("LevelOneTrigger") && PuzzleRoomOne.puzzleRoomOneComplete == false) // If the colliding object has tag 'LevelOneTrigger'
    {
        sceneController.LoadLevel("puzzleRoomOne"); // Load the scene by passing its name, on collision
    }
    #endregion
```

*Figure 39: Base scene, exemplify empty game objects with collider attachments, configured as triggers, for the use of scene transitioning. Illustrate box collider component attached to door object, for preventing player object from transitioning to other game scenes. 'PlayerController' class implementation of the 'OnTriggerEnter' method, shown in the bottom region of the figure*

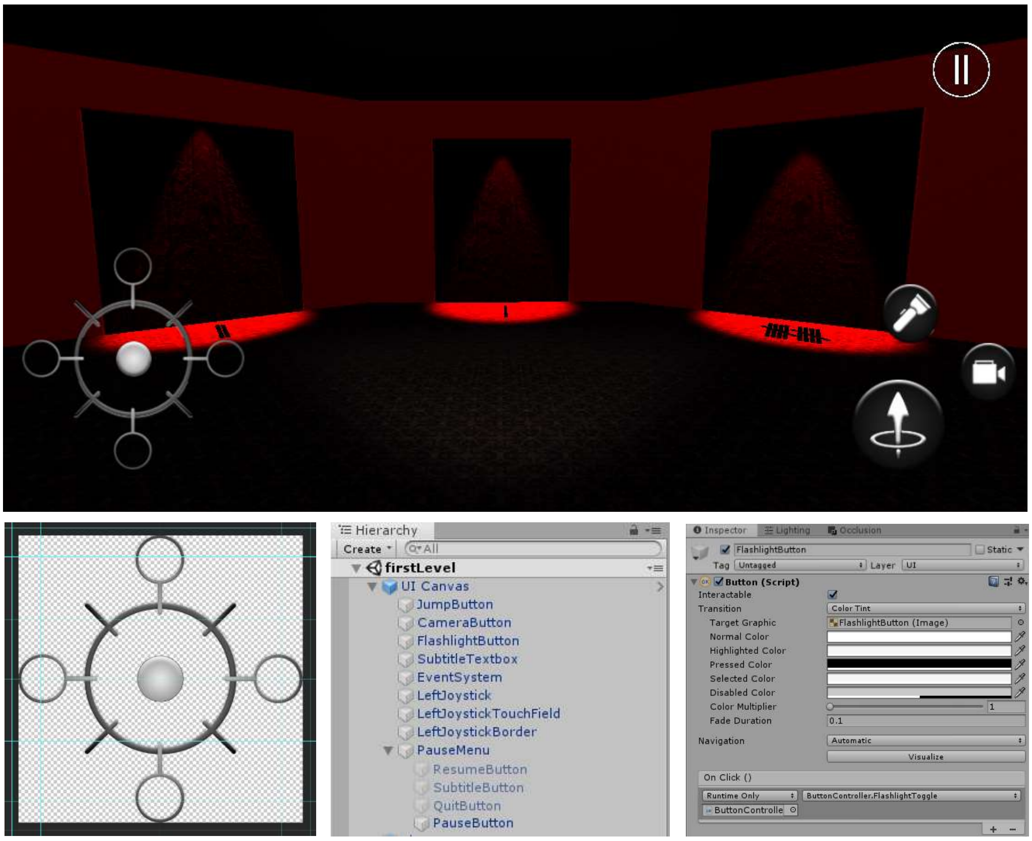
Supplementary to the games design, the functional and aesthetical[70] implementations of the pause menu and in-game UI's, were prioritised for adhering to the fundamental user requirements of the game, and for being a necessity to testing procedures. For which, a UI canvas object was created for parenting a series of button, text, and image UI objects[71]. The functionality that was intended for each of these elements[72], was designated upon completing its preliminary implementation[73].

[70] From the use of the image manipulation software application, Adobe Photoshop.

[71] Said objects possess image, text, button, and event trigger components for addressing each interfaces functional requirements, as projected within the functional requirements document (see ***Appendix B***).

[72] This functionality was composed up of the 'FlashlightToggle', 'CameraToggle', 'PauseGame', 'ResumeGame' and 'ExitGame' methods, which reside within the 'ButtonController', 'WindowGUI' and 'SceneController' classes (see ***Appendix H***). Additionally, the implementation of the 'LeftJoystickHandler' and 'PlayerController' classes were advanced, relative to the player movement and player animation-based methods.

[73] Assigning scripts to the button elements of the UI, required the passing of the scripts to their button components 'OnClick' field. This allowed individual methods of the scripts class, to be selected and invoked upon click-based interaction.



```
Hierarchy
Create ▾  Q▾All
▼ firstLevel
  ▼ UI Canvas
      JumpButton
      CameraButton
      FlashlightButton
      SubtitleTextbox
      EventSystem
      LeftJoystick
      LeftJoystickTouchField
      LeftJoystickBorder
    ▼ PauseMenu
        ResumeButton
        SubtitleButton
        QuitButton
        PauseButton
```

```
Inspector   Lighting   Occlusion
☑ FlashlightButton                          □ Static ▾
Tag  Untagged          Layer  UI
▼ ☑ Button (Script)
Interactable          ☑
Transition            Color Tint
  Target Graphic      FlashlightButton (Image)
  Normal Color
  Highlighted Color
  Pressed Color
  Selected Color
  Disabled Color
  Color Multiplier                         1
  Fade Duration       0.1
Navigation            Automatic
                      Visualize
On Click ()
Runtime Only          ButtonController.FlashlightToggle
ButtonControlle
                                             + −
```

*Figure 40: Pause menu and in-game UI designs, displaying UI canvas object hierarchy in relation to its UI elements. Aesthetic design process of UI elements within Adobe Photoshop is shown. Script passing to UI button elements is illustrated also.*

## Second Puzzle Room

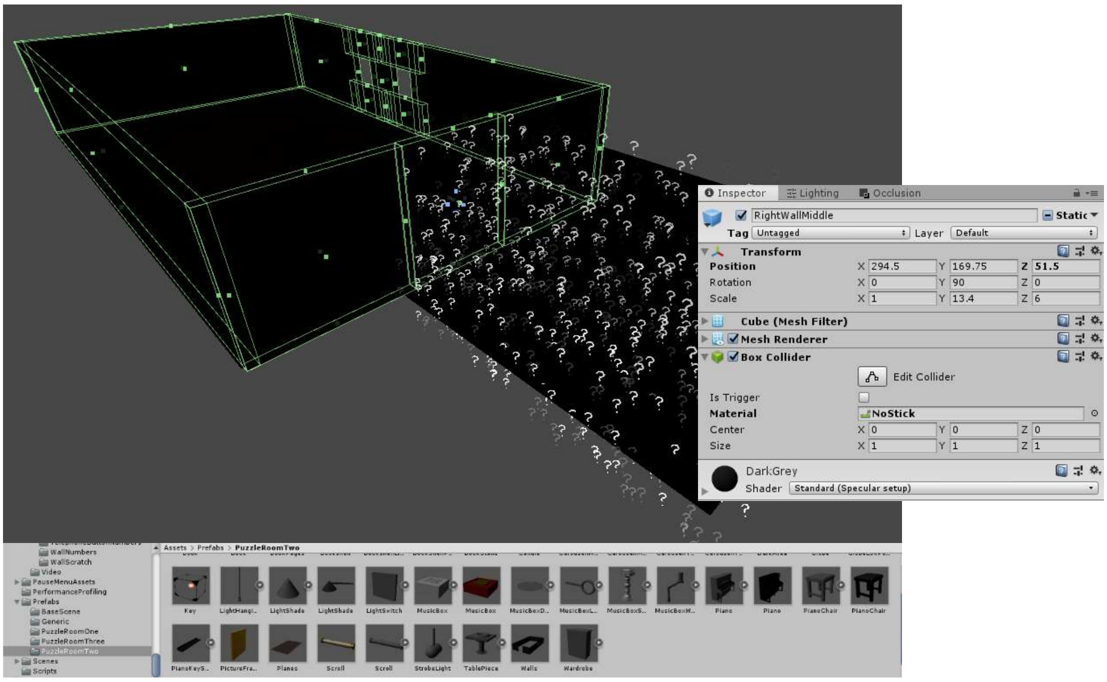
Meanwhile, for the initial development of the second puzzle room, a series of planes and cube objects were arranged into the formation of a box[74]; this layout provided a simple basis for a room-like environment[75]. Each of the objects were given a material, and collider component to prevent the player-controlled object from leaving the playable area[76]. Accompanying this, the door and dark area objects from the base scene were prefabricated and recycled, for time preservation purposes[77]; as were the UI canvas and player objects.

[74] The planes where purposed for the ground and ceiling surfaces of the room, whereas the cube objects were used as walls to border the environment.

[75] Which is a fundamental design concept for the escape-the-room orientation of the game.

[76] This use of collider components has been adapted for addressing a functional requirement of the game, to prevent a player from leaving the intended area of play, as detailed within the functional requirements document (see ***Appendix B***).

[77] Prefabricated objects employed the same materials, components, and configurations.



Inspector | Lighting | Occlusion

☑ RightWallMiddle — Static

Tag: Untagged — Layer: Default

Transform
- Position: X 294.5, Y 169.75, Z 51.5
- Rotation: X 0, Y 90, Z 0
- Scale: X 1, Y 13.4, Z 6

Cube (Mesh Filter)

☑ Mesh Renderer

☑ Box Collider
- Edit Collider
- Is Trigger: □
- Material: NoStick
- Center: X 0, Y 0, Z 0
- Size: X 1, Y 1, Z 1

DarkGrey
- Shader: Standard (Specular setup)

*Figure 41: Puzzle room two, showcasing the room-like environment, constructed from a series of cube and plane objects. Collider components are visualised by the green borders of geometry and within the inspector panel, alongside the material equipped by a wall object*

### *Safe Puzzle*

Fundamental to the safe puzzle, touch, look, and swipe-based interactions[78] in combination with audio dynamics[79], were implemented as the puzzle's mechanical focus[80]; which govern the rotary procedures for the safe dial object[81], and the resulting iteration of the puzzle[82]. For the performance of these interactions, camera mechanisms[83] were also addressed for stimulating an interaction view[84]. Said view could be incited by the player, through touch-based interactions with puzzle objects[85].

[78] Core focuses of the functional requirements (see ***Appendix B***), and literature review documents {see ***Appendix A***).

[79] Audio is reactively invoked by player interactions.

[80] The safe puzzle presents a trial-by-error focus for solving, which aims to heighten cognitive learning processes, through repetition.

[81] Rotary mechanics of the safe dial object are subject to the direction and type of swipe interaction performed. As well as being factored by the current iteration of the puzzle, determining the direction that safe dial object can be turned. Audio cues are invoked, to identify the correct numbers in the combination.

[82] Each iteration of the safe puzzle is represented by each number correctly navigated to, in the combination.

[83] Camera applications are detailed in the ***Functional Requirements*** section.

[84] The requirement for object interaction views is detailed within the functional requirements document (see ***Appendix B***).

[85] Picture canvas, safe dial, and safe door objects; all of the objects referred to were modelled within the three-dimensional modelling software application, Autodesk Maya.

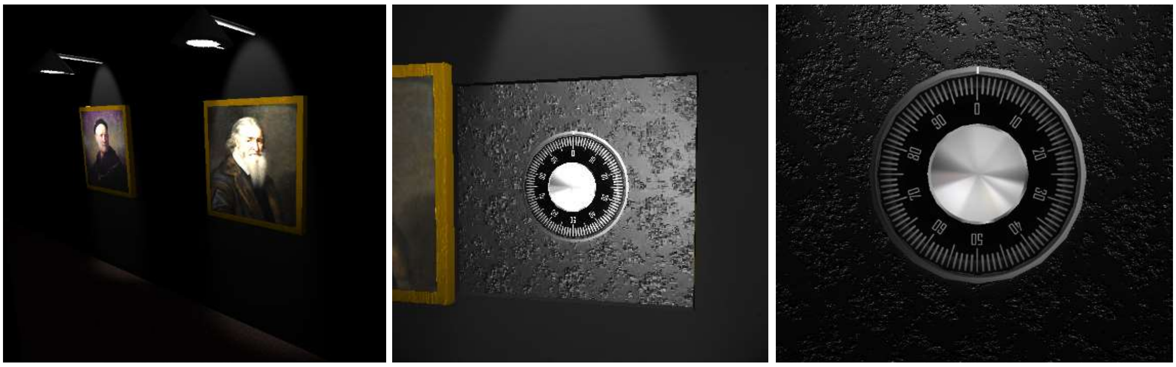


*Figure 42: Puzzle room two, hanging light object above the picture canvas object, safe object, and the safe interaction view perspective (captioned left to right)*

The puzzles dependence on audio dynamics, is purposed to challenge players unconventionally[86]; which is intended for slowing the pace of gameplay and promoting cognitive learning. This principle is presented by the puzzles fail-safe design[87], which is administered by a range of methods within the 'PuzzleRoomTwo' class[88]. For introducing the puzzle to the player[89], a combination of audio and animation components were assigned to a hanging-light object; which was strategically fitted, for implying the initial interactions of the scene[90].

### *Table Piece Puzzle*

Similar in mechanical outlooks as the safe puzzle, the table piece puzzle utilises touch, look, and swipe-based interactions[91], audio cues and camera mechanisms, for authoring puzzle establishment[92] and developments within the scene's narrative[93]. Alongside the functionality specified, the puzzle also integrates a multidirectional rotary mechanic[94], and a picker mechanic[95], for its globe and country marker objects. This integrates challenge[96], through exercising a player's ability to reminisce and reproduce animated sequences[97], interactively.

[86] This was detailed within the literature review (see ***Appendix A***), for the purpose of being dissimilar to existing game objectives in the genre.

[87] The puzzle caters for repetition and error conditioning, which is factored by the safe dial object being rotated in the opposing way intended for a given puzzle iteration.

[88] For the safe puzzle logic and implementation specifics, refer to ***Appendix I***.

[89] As a source of information, offered to the player for assisting progression. This is conceptualised within the literature review (see ***Appendix A***).

[90] The audio source equipped three-dimensional spatialization effects for enabling players to locate the source of the sound, whilst the animation state entailed light component flickering, to visually reinforce the sound effect being played. The hanging light object also positions above the interactable objects, to exhibit interaction.

[91] Swipe interactions for the table piece puzzle adopt a touch-drag orientation.

[92] Look-based interactions with the table piece stand is applicated to invoke hanging light animation states, that identify and invoke behavioural support, for puzzle interactivity.

[93] Authored by non-linguistic audio cues, as a mandatory feature of the game, discussed within the functional requirements document (see ***Appendix B***).

[94] Applied to the table globe object, for enabling non-visible country marker objects, to be translated into the viewport of the camera.

[95] Applied to the country marker objects, for addressing selective sequencing.

[96] Discussed within the literature review document (see ***Appendix A***).

[97] Players are expected to observe an animated sequence of material-lit country marker objects, and to then reproduce the sequence, via touch and swipe-gesture interactions with the table globe and country marker objects.

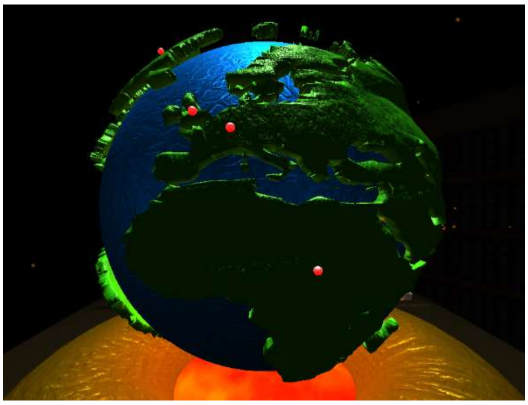


Figure 44: Puzzle room two, table piece puzzle, displaying table globe interaction view and arrangement for the country marker objects (material-lit)

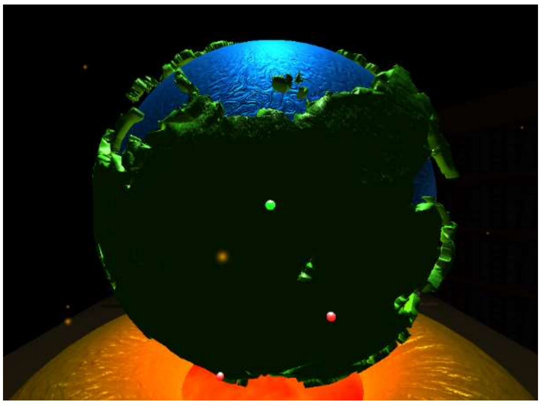


Figure 43: Puzzle room two, table piece puzzle, showcasing the first puzzle iteration, animation sequence

In further correspondence to the challenge presented, the puzzle issues a series of object-animated sequences, that incline in length and interactive requirement, for every iteration of the puzzle that the player reproduces correctly[98]. Said sequences and interactive engagements are supplied by the presence of object animator, audio source and collider components; which have behavioural implementations residing in the 'PuzzleRoomTwo' class[99]. Additional to the implementation discussed, the puzzle also adheres to a fail-safe design, for addressing puzzle resequencing and cognitive learning focuses[100].

### Music Box Puzzle

Essential to the music box puzzle, touch, look, swipe, pinch[101], and pinch-rotate[102] interactions were incorporated[103], for addressing the assembly procedures of the music box[104], engaging the picture frame, initiating carousel music box and hide cinematics[105], and for obtaining the catalogue of music box items[106]. To captivate the realism and immersion prospects of said interactions[107], a combination

[98] For every iteration of the puzzle that is reproduced correctly, the following sequence expands by one entry.

[99] For the table piece puzzle logic and implementation specifics, refer to ***Appendix J***.

[100] For the sequencing orientation of the puzzle, a player's ability to remember orders can develop (see ***Appendix F***).

[101] Pinch inwards and outwards gestures are used to invoke the 'Z' dimensional translation of the first-person camera, within the music box interaction view.

[102] Implemented to visualise a rotary offset for objects, which feature a rotary-basis of interaction.

[103] For their visualisation, see ***figure 19***.

[104] The puzzle fundamental objective is to assemble the music box, using the music box items obtained in previous developments of the puzzle.

[105] The picture frame cinematic purposes as a sinister development in the narrative, that characterises the torturous traits of the entity; the carousel music box cinematic purposes as a jump scare sequence, that introduces a unique method for obtaining puzzle items, and for heightening the sinister theme projected for the game; the hide cinematic purposes as a narrative development for the game, that amplifies suspense and plot interest.

[106] Music box, music box disc, music box lid latch key and the music box spindle wind up key; all of the mentioned objects were modelled within Autodesk Maya.

[107] Conceptualised within the literature review document (see ***Appendix A***).

of camera mechanisms[108] were supplemented to the range of object interaction perspectives[109], for stimulating player-controlled camera translation throughout the puzzle[110].

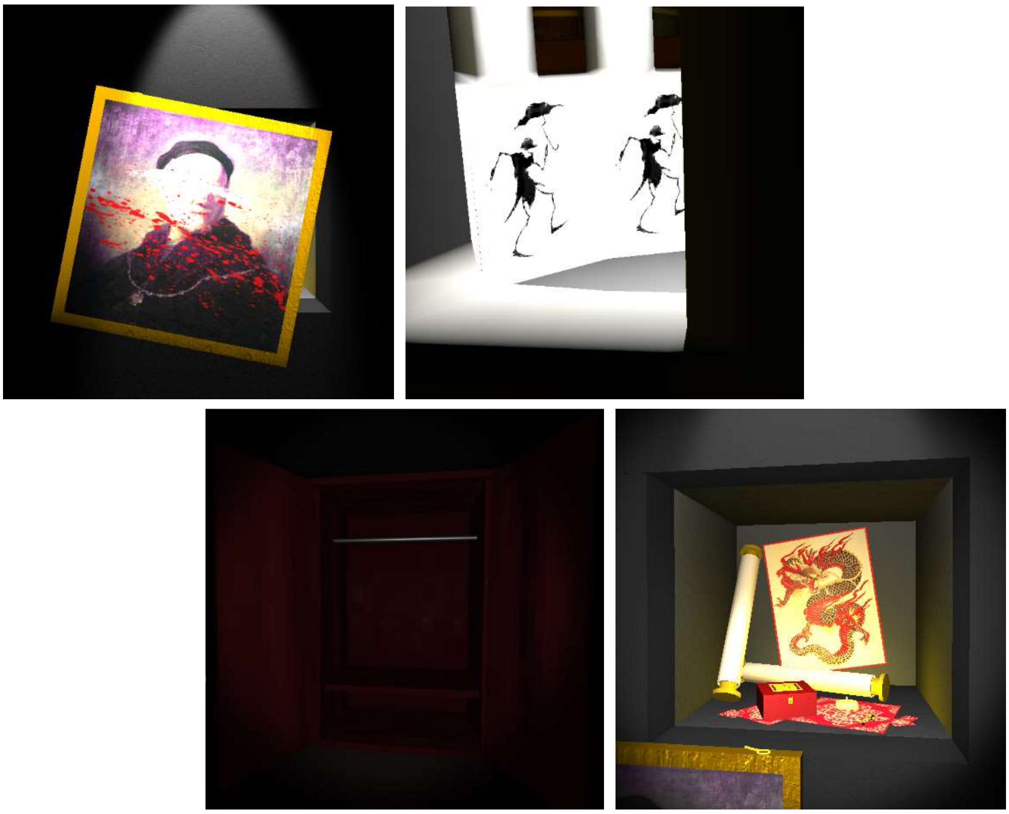


*Figure 45: Puzzle room two, music box puzzle, showcasing the picture frame, carousel music box and wardrobe cinematic sequences, alongside the collection of music box items that can be obtained (captioned left to right)*

As the fundamental basis of the puzzle, interaction sequences were complemented by a series of animations and audio cues, for identifying the progress state of each interaction[111], and for addressing the cyclical mechanisms of incomplete interactions[112]; which are functionally implemented within the 'PuzzleRoomTwo' class[113]. For authoring challenge, players are expected to discover and acknowledge the variety of interaction methods[114], used to manipulate the position and rotation offsets of objects, for the sequence of interactions presented[115].

[108] Multidirectional camera movements in the form of panning, camera position translation for simulating zoom movements, and the prevalence of entering and exiting object interaction views.

[109] A series of interaction views were implemented for isolating and invoking the music box item interactions.

[110] Camera translation was necessary for supporting touch-based interactions, on a range of mobile device interfaces that each vary in screen size and responsiveness.

[111] Conveyed through the invocation of sounds that reinforce the visualisation of object-animated alignment and translation sequences.

[112] Incomplete interactions can be reattempted at any point, for allowing players to attempt other object interactions that are familiarised interaction techniques.

[113] For the music box puzzle logic and implementation specifics, refer to ***Appendix K***.

[114] Aimed at the development of players investigative abilities (see ***Appendix F***).

[115] Object interactions are not typically sequenced but some interactions are required to be complete, before other object interactions can be invoked.

### *Piano Puzzle*

Fundamental to the piano puzzle, touch, look and swipe-based interactions, were functionally expanded and reapplied[116] in combination with camera mechanisms[117]; for authoring the puzzles interactive development focuses, upon a book[118], bookshelf[119], bookstand[120] and an array of piano objects[121]. Said developments were supplemented with animations and audio cues, for visual reinforcement purposes[122]. For the puzzle's main interaction[123], a series of cyclical mechanisms were integrated, for addressing error-handling within the puzzles time-driven, yet sequenced-based design[124]; this enables the puzzle to be reattempted, as part of a cognitive process[125].

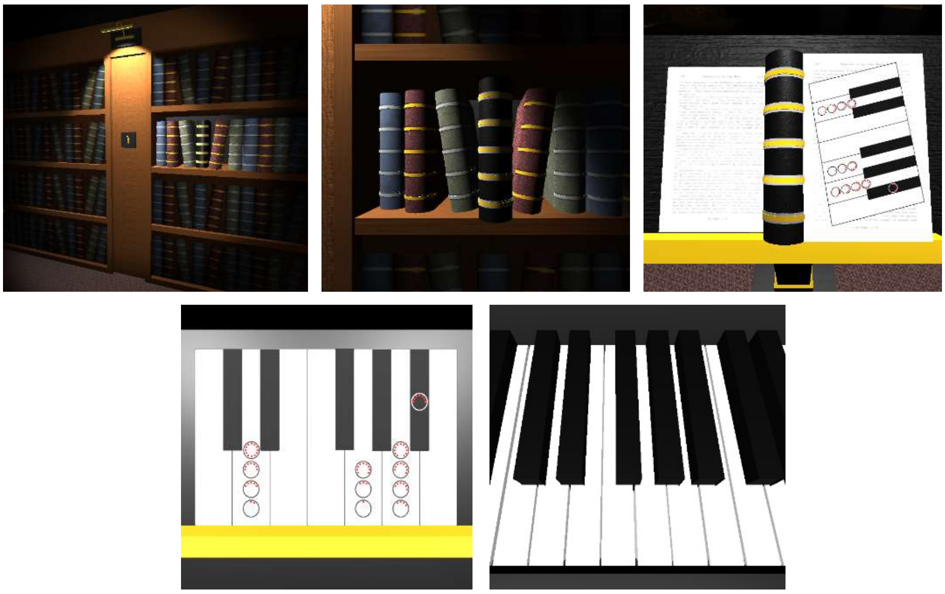


*Figure 46: Puzzle room two, piano puzzle, showcasing the book, bookshelf, bookstand, and piano object interaction developments*

Relating to the significance of time, a sequence of musical notes[126] must be reproduced by players, through a series of piano key interactions[127], which independently occur within five second intervals[128]. Time is factored as a pressure-mechanic, which is used to default puzzle progression, if

[116] Swipe-drag gestures were readapted, for allowing objects to be gradually translated, parallel to the moving motion of device peripherals and touch input.

[117] Recurring use of techniques to enter and exit object interaction views.

[118] Can be obtained, placed, and used to extract the music note sheet from its pages.

[119] Fundamental to the puzzle's initiation, a light switch embedded within the bookshelf can be interested with to allow the interaction with the book object, upon the puzzle-specific light becoming active in the scene.

[120] Designed as a placeholder to extract the music note sheet from its pages.

[121] Piano shelf and piano key objects; all of the objects presented were modelled within Autodesk Maya.

[122] Audio cues and animations were significant to the piano key objects, which were used to identify the keys being pressed and for enabling players to remember components of the sequence, auditorily. Upon the puzzle being reattempted.

[123] Piano key interactions.

[124] The puzzle requires players to reproduce the music note sequence, as an interrupted interaction.

[125] As conceptualised within the literature review document (see ***Appendix A***).

[126] Provided in a texture applied to a plane's material. This devises information offering for assisting puzzle progression, as discussed within the literature review document (see ***Appendix A***).

[127] Each piano key object attaches a animation and audio source component, to inform the interactions.

[128] A piano key must be interacted with, for every five seconds passed, or the puzzle progression will be reset.

the conditions are not adhered to[129]. The functional implementations of the puzzle's mechanisms are specified within the 'PuzzleRoomTwo' class[130].

## First Puzzle Room

Relating to the initial development of the first puzzle room, prefabricated objects created in the second puzzle room and base scenes[131] were reimported; for addressing consistency throughout the games aesthetical and mechanical configurations[132]. However, as the levels design deferred from inter-wall storage spaces[133], the composite of the 'walls' prefabrication needed updating[134]. Alongside this alteration, the tags that the objects were defaulted to, were also replaced for being irrelevant to the room, and for appropriating functional invocation and conditioning within class methods[135].

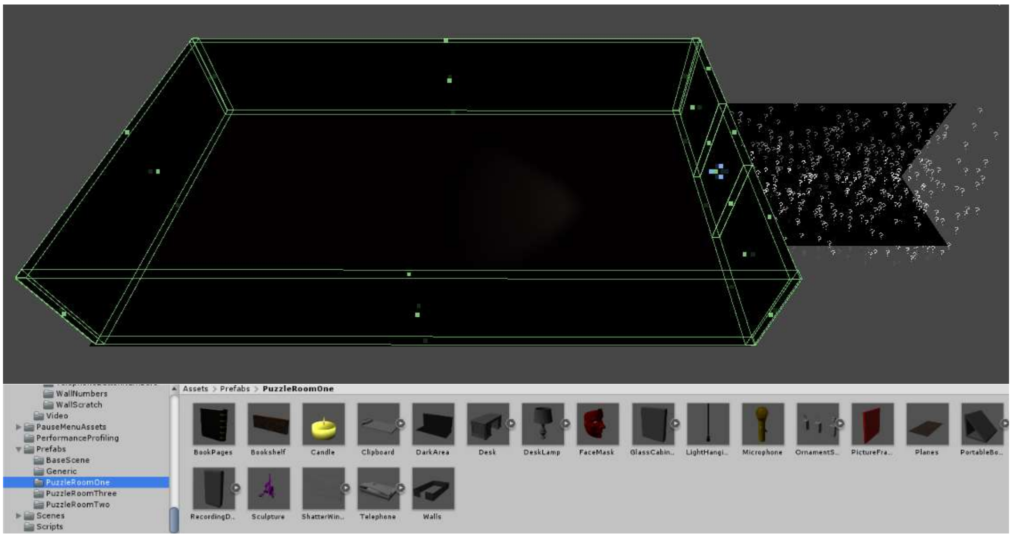


*Figure 47: Puzzle room one, room layout, featuring the walls, planes, door, and dark area prefabricated objects of the scene*

### *Telephone Puzzle*

As the first and only mechanic-aggregate in the scene[136], the telephone puzzle incorporates the use of camera mechanisms[137], touch, look, and swipe-based interactions[138], as well as audio cue

[129] Requiring players to reattempt the piano key sequencing element of the puzzle.

[130] For the piano puzzle logic and implementation specifics, refer to ***Appendix L***.

[131] This included the user interface canvas, player, door, dark area, walls, and plane objects.

[132] This process was hugely significant to sustaining the pace of development also, as opposed to recreating existing assets of the game.

[133] Used by the second puzzle room, as the table piece puzzle key storage, and music box puzzle part storage.

[134] A series of cubes and planes used to depict wall spaces, that were covered by picture canvas objects.

[135] This was necessary for preventing the player-controlled object from being able to progress to other scenes, without completing the puzzle room interactions. This would alert a conditioning defect within the 'OnTriggerEnter' method, in the 'PlayerController' class (see ***Appendix H***), if not addressed prematurely.

[136] The first puzzle room was designed to demonstrate the interconnectivity to the main development: the second puzzle room. This was also purposed for showcasing the more-skilled implementations of gameplay mechanics, through later progressions in the game.

[137] Camera applications are detailed in the ***Functional Requirements*** section.

[138] Said interactions are mentioned in the ***Functional Requirements*** section and originate within the functional requirements (see ***Appendix B***), and literature review documents {see ***Appendix A***).

invocation, to address challenge and narrative developments[139] throughout the scene. The fundamental objectives of the puzzle utilise the interaction types listed, for interacting with a series of objects[140], that each reveal a unique segment of a telephone number[141].

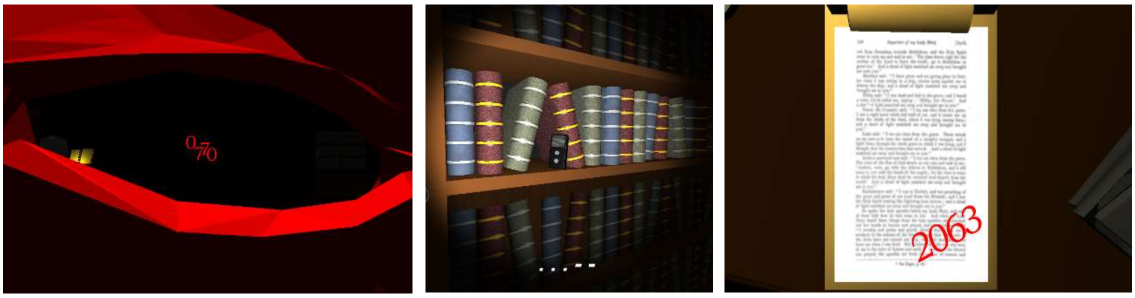
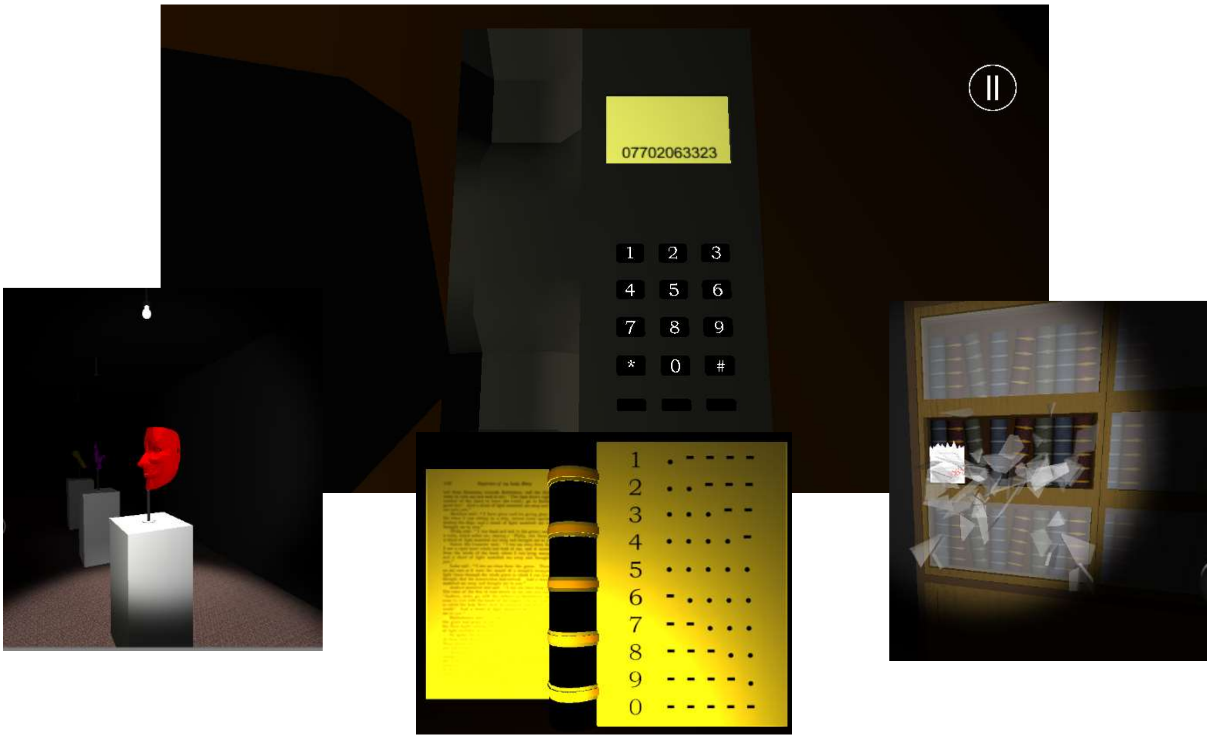


*Figure 48: Puzzle room one, showcasing the appearances of number segments, through object interaction views and audio cue invocation, which is assisted by subtitles when enabled. Face mask, audio device and clipboard object touch-based interactions (captioned left to right)*

This number sequence presents mechanical significance to the telephone object[142], which can be operated[143] to identify the correct order of the number segments[144]; cyclic mechanisms that allow the telephone object to be re-dialled, are invoked upon the incorrect number sequence being submitted[145]. Necessary for the puzzle's progression, the placement of animations, audio-to-text translations[146], and audio cues are used for offering information to players[147]; the application of these engagements is managed by the series of 'PuzzleRoomOne' class methods[148].

[139] Mentioned within the literature review document (see ***Appendix A***).
[140] Ornament stand, book, audio device, glass cabinet, and a note accompanied by a clipboard; all of the objects mentioned were modelled within Autodesk Maya.
[141] The telephone number is addressed as an integer array, which is used within the 'DialTelephone' method for comparing the telephone button interaction input. Seen in the 'PuzzleRoomOne' class (see ***Appendix H***).
[142] As the main feature of the scene.
[143] Subject to the puzzles progress state, which is managed within the 'TelephonePuzzleSolving' method, as seen in the 'PuzzleRoomOne' script (see ***Appendix H***).
[144] This is the underlying aim of the puzzle room components.
[145] These mechanics purpose for conditioning error, said conditioning can be found within the 'DialTelephone' method, located in the 'PuzzleRoomOne' class (see ***Appendix H***).
[146] Addressed by the presence of three-dimensional objects that employ materials, which equip illustrative-based textures. Said objects and textures were modelled and produced within Autodesk Maya, and Adobe Photoshop software applications.
[147] This technique is referred to within the literature review document (see ***Appendix A***), and the ***Background*** section.
[148] For the telephone puzzle logic and implementation specifics, refer to ***Appendix M***.



1 .----
2 ..---
3 ...--
4 ....-
5 .....
6 -....
7 --...
8 ---..
9 ----.
0 -----

*Figure 49: Puzzle room one, demonstrating the telephone object interaction state, as well as the series of engagements for offering players information. Hanging light animation, Morse code translation and glass cabinet window animation (captioned left to right)*

## Development Adherence

Leading the development of the application, and the production of the projects supporting documentation, the agile software development framework, SCRUM, was integrated as the project's development scheme. This framework was targeted for its incremental delivery of application behaviours, through segmenting its functional requirements into 'sprints'; which are designed for enhancing the productivity of the projects development, and the delivered quality of its products [31][149].

Accompanying the exploratory testing-basis for the applications behaviours, the incremental-development focus of the framework, enabled application functionality to be refactored quickly, as code base expansions were supplemented to the application in gradual, manageable volumes. This process further facilitated test-driven development, which assisted with the identification and refactorization of behavioural abnormalities, across the back-end and front-end architectures of the application. As well, alterations and extensions anticipated for the applications design, during its development cycle, could be addressed quickly[150].

In preparation of the project's undertaking, a Gantt chart representing the time investment, and resource requirements for completing the project was assembled; the existence of said chart was functionally significant to the development plan of the project[151]. Though the chart does not detail any of the intricate requirements for the projects components, it fulfils its ability to instruct the

[149] The projects adherence to the development scheme aforementioned, was initially introduced within the system design document (see ***Appendix D***).

[150] This ability was presented as a flexible characteristic of the framework [31].

[151] As a timeline of measures needing to be addressed.

projects order of development, and to dictate a series of time periods that each element of the project should be delivered within.

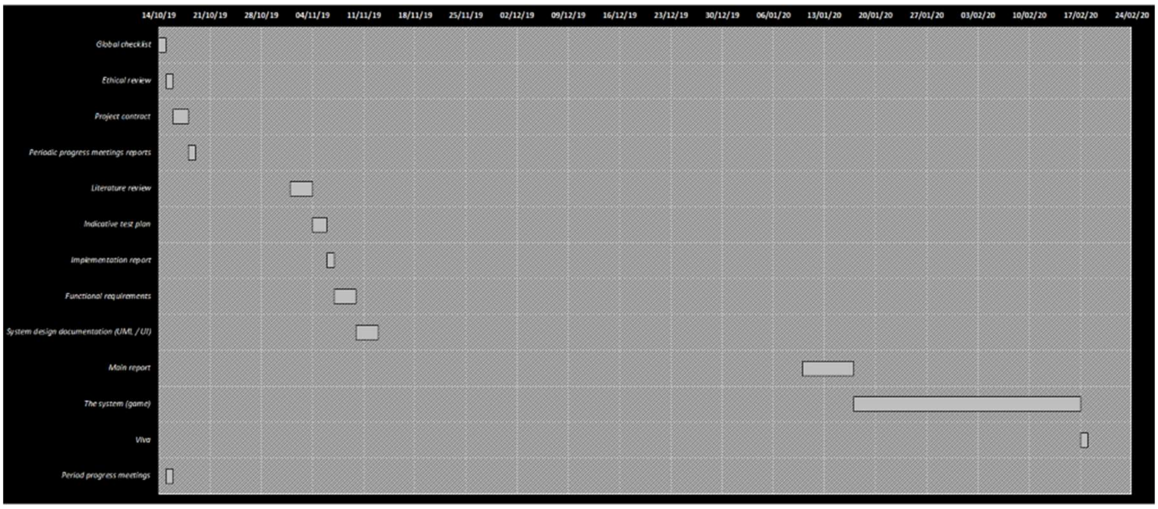


*Figure 50: Gantt chart, displaying the projects preliminary expectations of time investment for the academic year*

Meanwhile, throughout and upon the accomplishment of the project's development, an additional Gantt chart was populated for illustrating discrepancies between the initial development plan formulated, and the development plan that was exercised. As revealed within the updated version of the timeline, the adherence to the delivery of most project components was defective; this became apparent over the development of the project, due to the underestimation of application function density, deferrals addressed to the project for overcoming difficulties in other modules being studied, extensions to the functionality originally proposed for the application, and for the additional time granted for the delivery of the project[152].

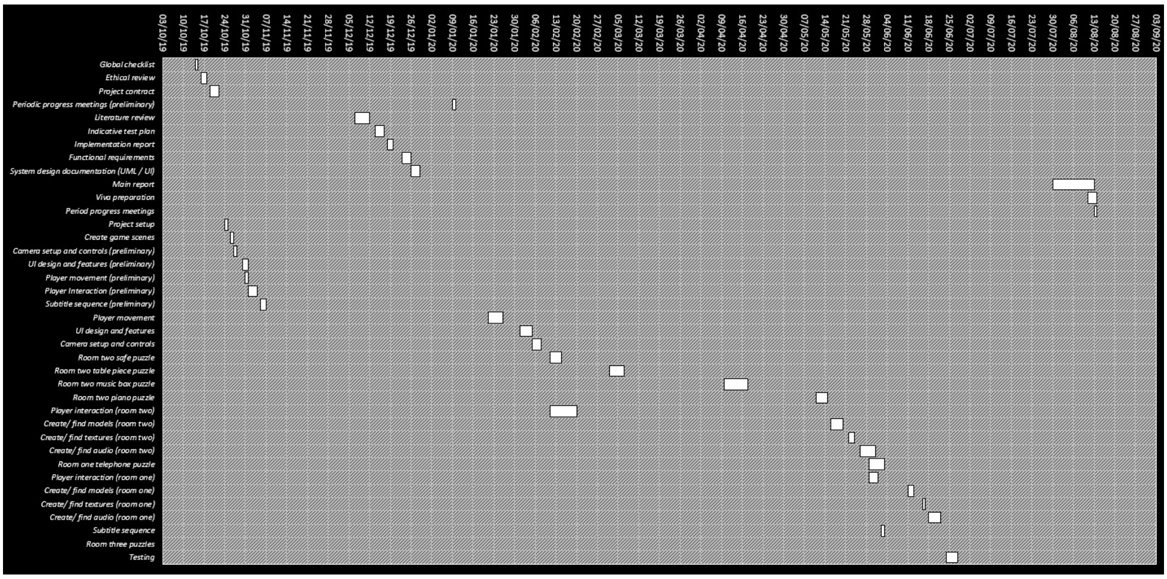


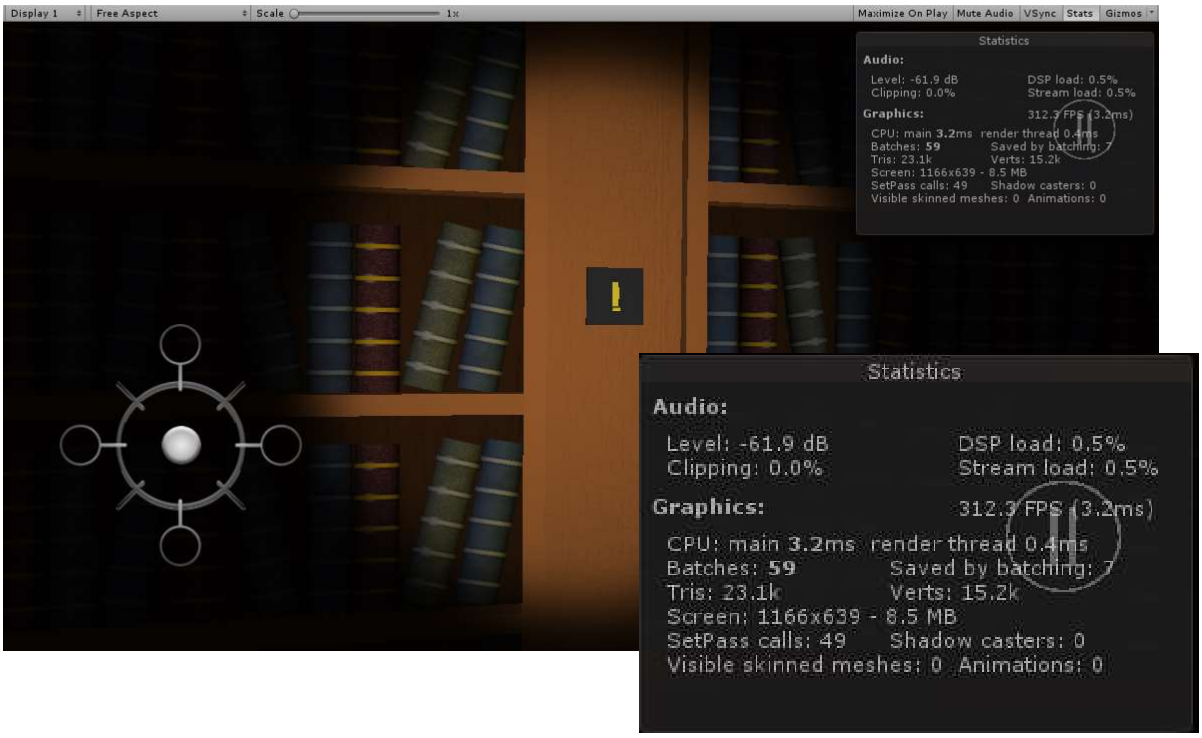
*Figure 51: Gantt chart, showcasing the time invested into the project's development for the entire span of the academic year, updated*

[152] As compensation for the development restraints caused by the social pandemic [32].

## Problem Resolution

Throughout the development cycle of the application, a range of programmatical, graphical and auditory defects were encountered, during testing and build procedures. As proposed within the indicative test plan[153] and ethical review[154] documents, improper functionality of the application would be revised, upon its discovery, and the games production values[155] would also aim to provide players with the best gameplay experience possible[156].

Discovered within the scenic developments of the second puzzle room, plunges in the number of frames rendered per-second were observed, from the statistics panel embedded within Unity's editor interface; this was identified as a compromise for importing object models with sophisticated geometry. In use of the tool, the number of vertices used to represent a model's primitive geometry could be acknowledged, on a per-object basis, when positioned within the active cameras' frustum[157].



```
Statistics
Audio:
  Level: -61.9 dB              DSP load: 0.5%
  Clipping: 0.0%               Stream load: 0.5%
Graphics:                      312.3 FPS (3.2ms)
  CPU: main 3.2ms  render thread 0.4ms
  Batches: 59        Saved by batching: 7
  Tris: 23.1k        Verts: 15.2k
  Screen: 1166x639 - 8.5 MB
  SetPass calls: 49    Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

Statistics
Audio:
Level: -61.9 dB DSP load: 0.5%
Clipping: 0.0% Stream load: 0.5%
Graphics: 312.3 FPS (3.2ms)
CPU: main 3.2ms render thread 0.4ms
Batches: 59 Saved by batching: 7
Tris: 23.1k Verts: 15.2k
Screen: 1166x639 - 8.5 MB
SetPass calls: 49 Shadow casters: 0
Visible skinned meshes: 0 Animations: 0

*Figure 52: Puzzle room two, low-poly bookshelf captured within the active camera's frustum, objects vertex count is show within Unity's statistics panel*

To combat graphical performance dampening, the three-dimensional graphics application, Autodesk Maya [33], was deployed as an initial effort for increasing the number of frames rendered per-second; this intention was addressed through reducing the number of polygons that objects employ.

[153] ***Appendix C***.

[154] ***Appendix G***.

[155] As a source of entertainment.

[156] Gameplay experience presents particular emphasis on device performance, as a leading influence on the applications usability.

[157] This adaptation of the tool was useful for isolating the models that impacted the graphical performance mostly.

For retargeting objects with lower polygonal counts, a percentage or count representing the polygonal reduction amount, could be passed in the reduction tools interface[158].

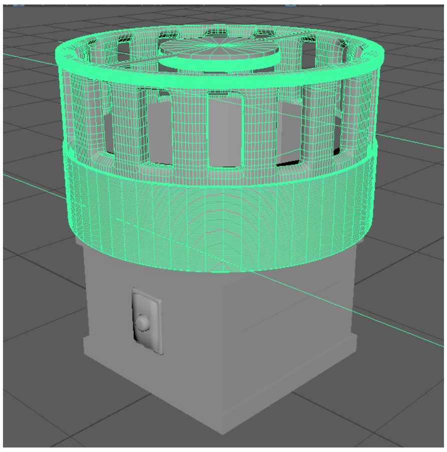
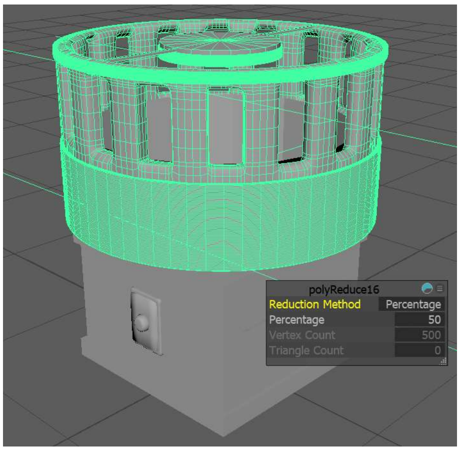


*Figure 53: Autodesk Maya, carousel music box object, showcasing the standard geometry count within the objects mesh*

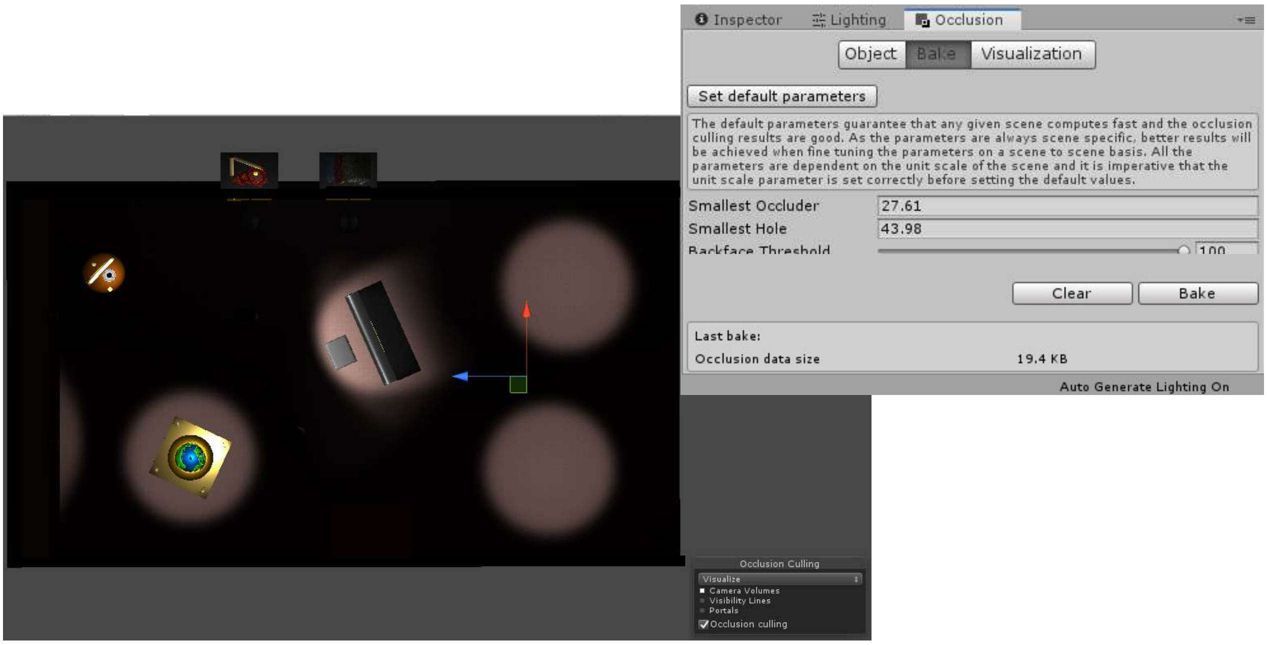


polyReduce16

| | |
|---|---|
| Reduction Method | Percentage |
| Percentage | 50 |
| Vertex Count | 500 |
| Triangle Count | 0 |

*Figure 54: Autodesk Maya, carousel music box object, displaying the reduced geometry count within the objects mesh. The mesh geometry reduction can be seen within the mesh reduction tools interface*

Alongside object polygonal reduction, occlusion culling was also configured in each of the Unity game scenes, for reducing the geometry count that is submitted for rendering; this is achieved by culling geometry in the scene, which the active camera cannot see [34], thus fewer draw calls are required for rendering the games environments. For its implementation, geometry culling data within each of the games scenes was baked, as featured in Unity's occlusion culling interface.



The default parameters guarantee that any given scene computes fast and the occlusion culling results are good. As the parameters are always scene specific, better results will be achieved when fine tuning the parameters on a scene to scene basis. All the parameters are dependent on the unit scale of the scene and it is imperative that the unit scale parameter is set correctly before setting the default values.

| | |
|---|---|
| Smallest Occluder | 27.61 |
| Smallest Hole | 43.98 |
| Backface Threshold | 100 |

Last bake:

Occlusion data size 19.4 KB

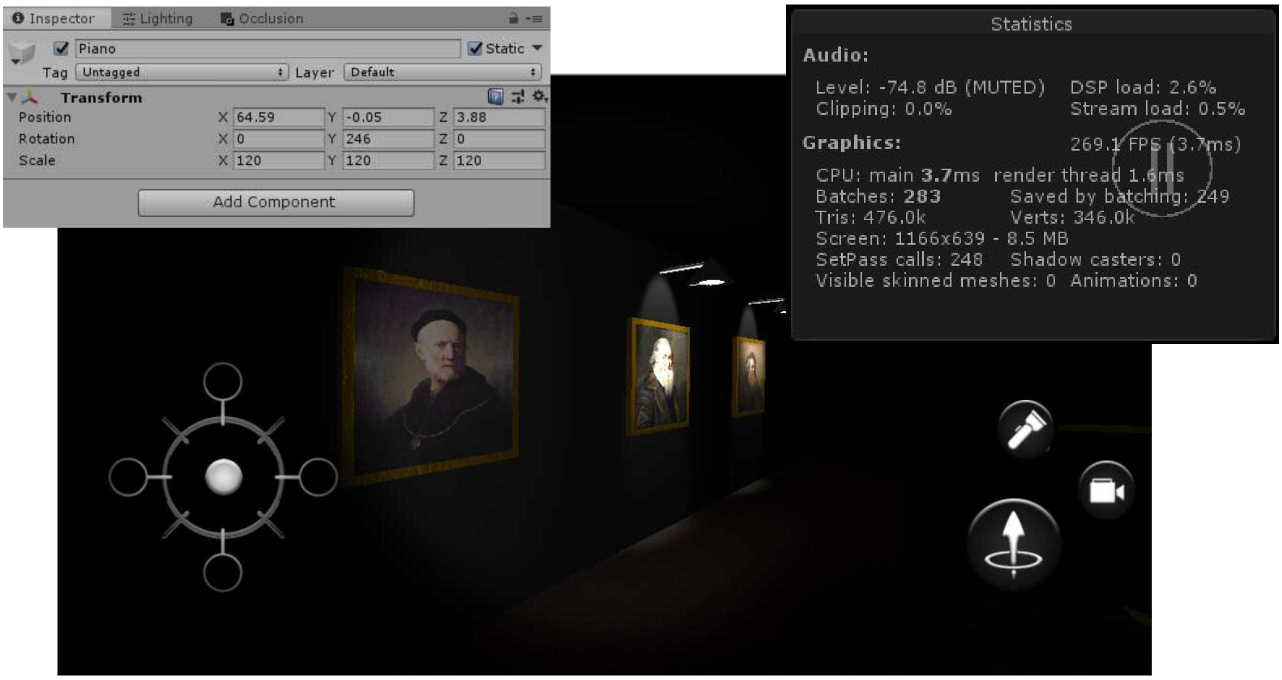
Auto Generate Lighting On

■ Camera Volumes
□ Visibility Lines
□ Portals
☑ Occlusion culling

*Figure 55: Puzzle room two, visualizing occlusion culling, unoccupied spotlight areas of the scene represent the areas where geometry has been culled. Occlusion culling configuration is seen within Unity's occlusion culling panel*

[158] This tool is located within Maya's mesh tool suite.

Also beneficial to the application's graphical performance, objects that were recognized as stationary, were registered as static; this allowed the meshes of statically-identified objects, to be combined for issuing fewer draw calls[159] [35]. To address static batch-rendering for individual game objects, the static field within Unity's inspector panel was selected; this feature was predominantly applied to the decorative objects of the game, that bear no functional significance, as aesthetic commodities.



```
☑ Piano                         ☑ Static
Tag  Untagged          Layer  Default
Transform
Position   X 64.59   Y -0.05   Z 3.88
Rotation   X 0       Y 246     Z 0
Scale      X 120     Y 120     Z 120
```

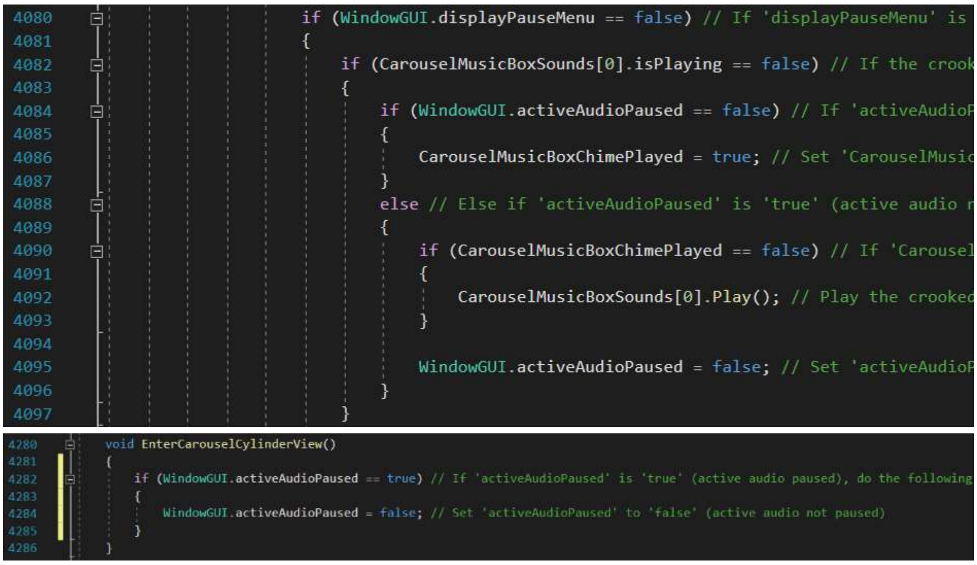
```
Statistics
Audio:
  Level: -74.8 dB (MUTED)    DSP load: 2.6%
  Clipping: 0.0%             Stream load: 0.5%
Graphics:                    269.1 FPS (3.7ms)
  CPU: main 3.7ms  render thread 1.6ms
  Batches: 283         Saved by batching: 249
  Tris: 476.0k         Verts: 346.0k
  Screen: 1166x639 - 8.5 MB
  SetPass calls: 248   Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

*Figure 56: Puzzle room two, displaying the number of draw calls saved by statically batching stationary objects, as seen within the statistics panel. Assigning game objects for static batching is illustrated within the inspector panel*

Moreover, posing as a programmatical hindrance, audio cues would unintentionally loop upon being invoked, and when the game session is paused prior to the first-person camera being stationed at an object interaction view. This behaviour was identified as a defect of method conditioning, which was combated by setting the static Boolean variable, 'activeAudioPaused', to its default state within each of the relevant methods[160].

[159] As seen within the adaptation of occlusion culling.

[160] Enter and exit object interaction view methods.



```
if (WindowGUI.displayPauseMenu == false) // If 'displayPauseMenu' is
{
    if (CarouselMusicBoxSounds[0].isPlaying == false) // If the crook
    {
        if (WindowGUI.activeAudioPaused == false) // If 'activeAudioP
        {
            CarouselMusicBoxChimePlayed = true; // Set 'CarouselMusic
        }
        else // Else if 'activeAudioPaused' is 'true' (active audio r
        {
            if (CarouselMusicBoxChimePlayed == false) // If 'Carousel
            {
                CarouselMusicBoxSounds[0].Play(); // Play the crooked
            }

            WindowGUI.activeAudioPaused = false; // Set 'activeAudioP
        }
    }
```

```
void EnterCarouselCylinderView()
{
    if (WindowGUI.activeAudioPaused == true) // If 'activeAudioPaused' is 'true' (active audio paused), do the following
    {
        WindowGUI.activeAudioPaused = false; // Set 'activeAudioPaused' to 'false' (active audio not paused)
    }
}
```

*Figure 57: 'PuzzleRoomTwo' class, setting 'activeAudioPaused' to its default state when entering the carousel music box cylinder object view*

Lastly, as part of the rotatory and translation mechanisms for the first-person camera[161], when entering and exiting object interaction views, the camera object would often not reach its target position and rotation offset; which would cause the game session to become dysfunctional, where players cannot proceed through the game[162].

For reducing the occurrence and impact of the problem, all of the applications functionality factoring delta time, was replaced with smooth delta time; this was better adapted for smoothing object animations as well as for reducing calculation oversights[163]. Also, the implementation of the camera's movement was refactored, for comparing the transforms of the camera and target, before the camera rotates or translates; this aimed to prevent the possibility of the camera object over-stepping the target transform, for a consecutive number of frames.

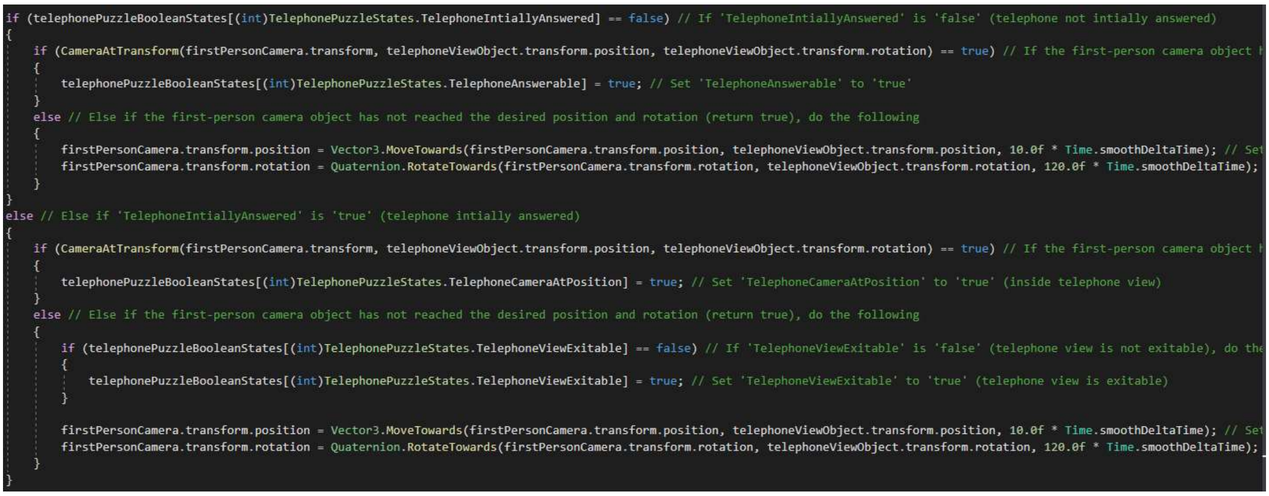
Regardless of the efforts delivered for understanding and resolving this error, it remains a hinderance to the quality of the game. Due to the mechanisms prevalence to function more than to fault[164], the defect could be considered a liability of the integrated methods deployed by the Unity engine.

[161] Camera applications are detailed in the ***Functional Requirements*** section.

[162] UI elements do not reappear, and object interactions cannot be invoked also.

[163] This was assumed to function better for averaging the movement of the camera over time, as opposed to the camera's movement being subjected to computational inconsistences.

[164] As realized from the regime of testing conducted for the game.



```
if (telephonePuzzleBooleanStates[(int)TelephonePuzzleStates.TelephoneIntiallyAnswered] == false) // If 'TelephoneIntiallyAnswered' is 'false' (telephone not intially answered)
{
    if (CameraAtTransform(firstPersonCamera.transform, telephoneViewObject.transform.position, telephoneViewObject.transform.rotation) == true) // If the first-person camera object
    {
        telephonePuzzleBooleanStates[(int)TelephonePuzzleStates.TelephoneAnswerable] = true; // Set 'TelephoneAnswerable' to 'true'
    }
    else // Else if the first-person camera object has not reached the desired position and rotation (return true), do the following
    {
        firstPersonCamera.transform.position = Vector3.MoveTowards(firstPersonCamera.transform.position, telephoneViewObject.transform.position, 10.0f * Time.smoothDeltaTime); // Set
        firstPersonCamera.transform.rotation = Quaternion.RotateTowards(firstPersonCamera.transform.rotation, telephoneViewObject.transform.rotation, 120.0f * Time.smoothDeltaTime);
    }
}
else // Else if 'TelephoneIntiallyAnswered' is 'true' (telephone intially answered)
{
    if (CameraAtTransform(firstPersonCamera.transform, telephoneViewObject.transform.position, telephoneViewObject.transform.rotation) == true) // If the first-person camera object
    {
        telephonePuzzleBooleanStates[(int)TelephonePuzzleStates.TelephoneCameraAtPosition] = true; // Set 'TelephoneCameraAtPosition' to 'true' (inside telephone view)
    }
    else // Else if the first-person camera object has not reached the desired position and rotation (return true), do the following
    {
        if (telephonePuzzleBooleanStates[(int)TelephonePuzzleStates.TelephoneViewExitable] == false) // If 'TelephoneViewExitable' is 'false' (telephone view is not exitable), do the
        {
            telephonePuzzleBooleanStates[(int)TelephonePuzzleStates.TelephoneViewExitable] = true; // Set 'TelephoneViewExitable' to 'true' (telephone view is exitable)
        }

        firstPersonCamera.transform.position = Vector3.MoveTowards(firstPersonCamera.transform.position, telephoneViewObject.transform.position, 10.0f * Time.smoothDeltaTime); // Set
        firstPersonCamera.transform.rotation = Quaternion.RotateTowards(firstPersonCamera.transform.rotation, telephoneViewObject.transform.rotation, 120.0f * Time.smoothDeltaTime);
    }
}
```

*Figure 58: 'PuzzleRoomOne' class, illustrating the functionality presented for controlling camera movement within the 'EnterTelephoneView' method*

## Testing Regime

### Overview

In accordance with the indicative test plan (ITP)[165], the applications development was intended to be tested in an exploratory nature[166], upon implementing sections of its code base. As an agile software testing methodology, exploratory testing has enabled the application to present more expected behaviours and functionality of its components, over the course of its development [37]; this has been achieved by refactoring the applications code base, which has been reactive to the observations and findings of the test cases conducted.

Relating to the categories of testing, unit, black-box and performance profiling were all considered for gauging the front-end and back-end implementations of the applications design; as outlined in the ITP, this regime aimed to advance the reliability and usability of the application. For which, the fundamental focuses of the test regime were upon the application's graphical complexity and mechanical composition. For the basis of testing, a vast amount of time has been deferred from test procedures for enhancing the software's development cycle[167], when compared to formal agile testing approaches [38]; which was possible by the problematic areas of the application being identified within the recent implementations of the software.

### Unit Testing

For the structural precision of the application, unit testing was nominated for determining the behavioural expectations of the application[168]. As mentioned in the ITP, testing by unit is effective for sourcing the arithmetical, conditional, and the resulting functional defects of an application, this is due to being able to isolate the values or states of variables and function members [39]. In this

[165] ***Appendix C***.
[166] A gradual and componentised approach to testing the software's development.
[167] Achieved by isolating programmatical defects upon finding.
[168] This was achieved by segmenting the applications code base, into components of functionality.

relation, it was sensible for selecting unit testing as an approach for identifying errors that may not be understood, observationally[169].

Adopting this strategy into the testing regime, has enabled the correctness of the applications functionality to be identified from a statistical standpoint[170]. For the test cases conducted, it was believed sensible to categorically unit test the application, as each script or controller would typically be developed independently of each other; allowing the regime for this testing approach to be more comprehensive.

## Black-box Testing

For testing the functional attributes of the application, black-box testing was accepted for identifying the correctness of the applications behaviours, from a visual standpoint; this allowed the test cases to be conducted simply and quickly, as opposed to the testing-by-unit approach. As programmatical involvement was not required, less time was consumed, which allowed time to be allocated to the applications development processes. Although black-box testing was not suitable for isolating programmatical errors, it was sensible for recognizing probable sources of error and inefficiencies; especially within input and output system relations [40][171]. The approach was also suited to exercising the applications user case scenarios[172].

In result of its employment, discrepancies within the applications functionality could be identified from an observational outlook; for this domain, it was well adapted for conducting test cases that are user case and state transition orientated [41]. Similar to unit testing, the black-box test cases were categorically conducted and sorted for comprehensive purposes.

## Performance Profiling

For testing system performance whilst executing the application, performance profiling was deployed for the purpose of identifying the applications influence on system performance, from a statistical perspective; this approach was necessary for measuring system resource consumption and identifying the subsequent stability and usability of the application. From sampling the performance of a system, profiling could be used to determine the segments of application code [42] and graphic properties, that contribute to degrading computational performance[173].

By utilising this approach, has enabled the application to be significantly more efficient over the course of its development; this is evident statistically and observationally. In accordance with the profiling and accompanying test cases conducted, it was conclusive that the complexity of model meshes being rendered, as well as the normal map, specularity and detail mask properties that their materials employed, were impacting the applications performance significantly. From these observations, it is inevitable that performance profiling was an appropriate strategy for identifying performance degradation.

[169] For the supporting unit test documentation, see ***Appendix N***.

[170] In focus of the exploratory testing method, programmatical errors could be erased upon their finding, which did not hinder future additions of functionality to the applications code base.

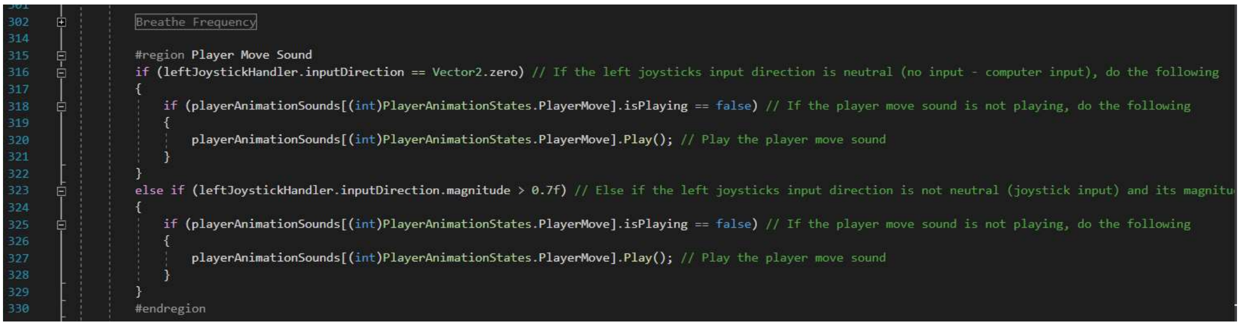
[171] As discussed within the indicative test plan (ITP) (see ***Appendix C***).

[172] As discussed within the software functional requirements document ***(see Appendix B)***; for the supporting black-box test documentation, see ***Appendix O***.

[173] For the supporting performance profiling documentation, see ***Appendix P***.

## Project Maintenance

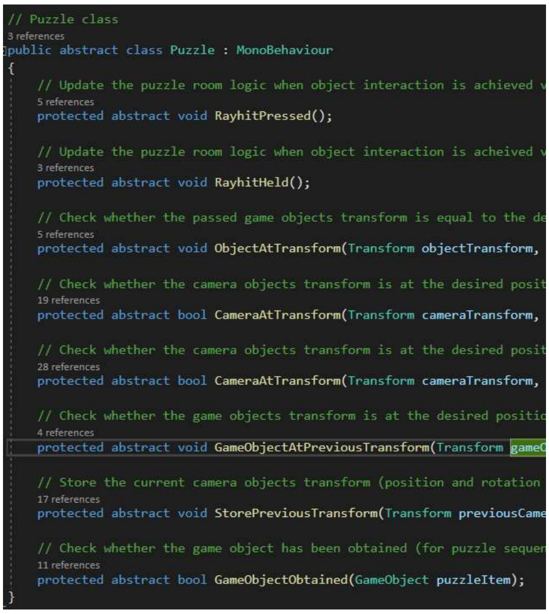
Throughout the development cycle of the project, the application code and accompanying documentation were categorically tasked and organised; this was intended for maintaining the projects rate of progress and comprehensive state. Programmatically, in use of Microsoft's Visual Studio Integrated Development Environment (IDE) [43], the applications code could be partitioned into regions and commented to contextualise the implementation at every line; this allowed the functionality of the code to be understood, without regards to the time of its addition to the code base. For such, regions and comments were populated parallel to the implementation.



```
Breathe Frequency

#region Player Move Sound
if (leftJoystickHandler.inputDirection == Vector2.zero) // If the left joysticks input direction is neutral (no input - computer input), do the following
{
    if (playerAnimationSounds[(int)PlayerAnimationStates.PlayerMove].isPlaying == false) // If the player move sound is not playing, do the following
    {
        playerAnimationSounds[(int)PlayerAnimationStates.PlayerMove].Play(); // Play the player move sound
    }
}
else if (leftJoystickHandler.inputDirection.magnitude > 0.7f) // Else if the left joysticks input direction is not neutral (joystick input) and its magnitu
{
    if (playerAnimationSounds[(int)PlayerAnimationStates.PlayerMove].isPlaying == false) // If the player move sound is not playing, do the following
    {
        playerAnimationSounds[(int)PlayerAnimationStates.PlayerMove].Play(); // Play the player move sound
    }
}
#endregion
```

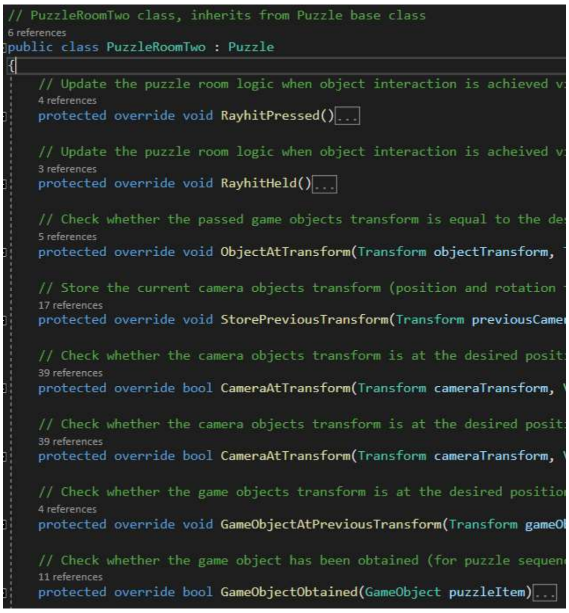
*Figure 59: 'PlayerController' class, showcasing code segmentation through the application of regions. Comments are also shown each line of implementation, these are led by the '//' parenthesis*

Additional to the tools provided by the environment, the preliminary Unified Modelling Language (UML) diagram[174], was implemented and referred to throughout the setup process of the applications development cycle; this provided the initial structure to the applications code base, which introduced OOP techniques[175]. This became apparent due to the existence, types, and relations between classes; however once implemented, alterations were made to the original design over time, to cater for the unforeseen functional and behavioural requirements of the application.



```
// Puzzle class
3 references
public abstract class Puzzle : MonoBehaviour
{
    // Update the puzzle room logic when object interaction is achieved v
    5 references
    protected abstract void RayhitPressed();

    // Update the puzzle room logic when object interaction is acheived v
    3 references
    protected abstract void RayhitHeld();

    // Check whether the passed game objects transform is equal to the de
    5 references
    protected abstract void ObjectAtTransform(Transform objectTransform,

    // Check whether the camera objects transform is at the desired posit
    19 references
    protected abstract bool CameraAtTransform(Transform cameraTransform,

    // Check whether the camera objects transform is at the desired posit
    28 references
    protected abstract bool CameraAtTransform(Transform cameraTransform,

    // Check whether the game objects transform is at the desired positio
    4 references
    protected abstract void GameObjectAtPreviousTransform(Transform gameO

    // Store the current camera objects transform (position and rotation
    17 references
    protected abstract void StorePreviousTransform(Transform previousCame

    // Check whether the game object has been obtained (for puzzle sequen
    11 references
    protected abstract bool GameObjectObtained(GameObject puzzleItem);
}
```

*Figure 61: 'PuzzleRoomTwo' class, illustrating the virtual methods being overridden as implementations of the interface. 'PuzzleRoomTwo' inherits from 'Puzzle' class*



```
// PuzzleRoomTwo class, inherits from Puzzle base class
6 references
public class PuzzleRoomTwo : Puzzle
{
    // Update the puzzle room logic when object interaction is achieved v
    4 references
    protected override void RayhitPressed()...

    // Update the puzzle room logic when object interaction is acheived v
    3 references
    protected override void RayhitHeld()...

    // Check whether the passed game objects transform is equal to the de
    5 references
    protected override void ObjectAtTransform(Transform objectTransform, T

    // Store the current camera objects transform (position and rotation
    17 references
    protected override void StorePreviousTransform(Transform previousCamer

    // Check whether the camera objects transform is at the desired posit
    39 references
    protected override bool CameraAtTransform(Transform cameraTransform, V

    // Check whether the camera objects transform is at the desired posit
    39 references
    protected override bool CameraAtTransform(Transform cameraTransform, V

    // Check whether the game objects transform is at the desired positio
    4 references
    protected override void GameObjectAtPreviousTransform(Transform gameOb

    // Check whether the game object has been obtained (for puzzle sequen
    11 references
    protected override bool GameObjectObtained(GameObject puzzleItem)...
```

*Figure 60: 'Puzzle' class, demonstrating the virtual method declarations, one of which functions are overloaded. 'Puzzle' class is abstract, an interface for the puzzle room classes*

[174] As seen within the system design document (SSD) (see ***Appendix D***).

[175] That were encapsulation, abstraction, inheritance, and polymorphism [44].

Meanwhile, in relation to the production of the testing documentation, the documentation was supplied simultaneously to the test cases being conducted, on the same systemic basis[176]; for which, the structure of the documentation resembles the categorised testing approach. Directing and documenting the test cases has enabled the testing regime to be better understood and well-adhered to.

# Critical Evaluation

# Project Evaluation

## System Features

Relating to the objectives and criteria addressed within the project contract[177], ethical review[178], global checklist[179] and functional requirements[180] documents, the delivered state of the system addresses all of the mandatory behaviours, through the series of mechanical, aesthetical, and auditory implementations presented. From an architectural standpoint, the application conforms to all of the OOP principles introduced by the programme, and also boasts a significant amount of functionality that has been used to address core and subsidiary components[181], for of range of uniquely-designed puzzles.

Given more time was available for the systems development, the first puzzle room would feature more puzzle components, as intended for the scene; which would entail the gyroscope capabilities of mobile devices[182], for the rotary mechanics of said components. Furthermore, the implementations for the camera mechanisms would be retargeted, for fulfilling its behavioural expectations when entering and exiting object interaction views; as it was not possible to address a stable solution[183], for the capacity of time offered. Additional to these applications, input-tracked gesture interactions could have been implemented, for the compatibility of intricate puzzle components[184], and for the delivery of more interesting interactions[185].

With regards to undelivered development, the functional requirements document[186] details the apparency of image fields within the in-game UI design[187]; which was purposed for displaying the items that the player obtains. Said feature was retargeted in the game, due to its insignificance in navigating puzzle progression, and for littering the cameras viewports[188]. Its absence from the application, was not a constraint of time nor complexity.

[176] As mentioned in the section prior.
[177] ***Appendix E***.
[178] ***Appendix G***.
[179] ***Appendix F***.
[180] ***Appendix B***.
[181] The main features of interaction for each puzzle, and the intermediate interactions necessary, to reach the point of main feature interaction.
[182] Physical rotations of mobile devices, as opposed to rotary interface interaction.
[183] Mentioned in the ***Testing*** section.
[184] Objects would be able to translate throughout the game world, relative to the position of interaction input.
[185] Discussed in the literature review (see ***Appendix A***).
[186] ***Appendix B***.
[187] Referred to in the system design document (SSD) (see ***Appendix D***).
[188] Referred to in the ***User Interfaces*** section.

# Development Evaluation

## Development Approach

For the preliminary instruction of the project's development, a Gantt chart was populated[189], for the immediate purpose of identifying a componentized format of the projects deliverable items[190]; this was then used to allocate an amount of time expected for each components delivery, and the dates that the components were expected to be delivered on. The preliminary design of the Gantt chart accounted for the incremental development framework that I had adopted for the project, SCRUM[191].

However, when reflecting upon the development approach that was exercised, the order, time investments and delivery dates for the project's components, greatly deferred from what was originally calculated. This was mostly factored by the developments of the second puzzle room scene, which subsequently become the initial puzzle development of the system; as opposed to the planned developments for the first puzzle room scene. Yet, as the second puzzle room scene envisioned a wider range of puzzle mechanics and aesthetic components, most of the scenes assets could be prefabricated and reimported into the first puzzle room scene; which benefit the systems rate of development, allowing for more puzzle components and accompanying functionality to be implemented, as opposed to what was originally anticipated.

Also, complementing the structural development of the system, a UML diagram addressing the back-end architecture of the application, was formulated[192], and adhered to within the initial setup and development processes of the game; which posed as the fundamental implementation for the system. However, throughout the later developments of the application, many classes, methods, and class members were redacted, refactored, or expanded upon, to cater for the additional behaviours desired for the game's scenes, and for overcoming the functional defects that were discovered within the applications testing procedures[193]. From the existence of the UML diagram, the applications mechanical expectations and developments could be quickly acknowledged, which enabled the implementation process to be hastened.

## Academic Advancement

For the role requirements of the projects undertaking, my understanding of managing software developments has been informed by the timing and delivery nature of the system, as are the procedures that are required to structure a development plan, for a game application; and to implement its front-end and back-end architectures, as a programmer.

Meanwhile, for the targeted production values of the game, I have successfully addressed a series of mechanically-driven puzzles, which engage a range of interactions that employ basic physical, numerical, and observational skill sets[194]; suitable for all cultures and environments that players are subjected to[195].

[189] Seen in the ***Development Adherence*** section.

[190] Generically addressed in the project contract document (see ***Appendix E***).

[191] Discussed in ***Development Adherence*** section.

[192] ***Appendix H***.

[193] Mentioned within the ***Problem Resolution*** section.

[194] Detailed within the ***Aims and Objectives*** section.

[195] Referred to within the global checklist document (see ***Appendix F***).

# Tool Evaluation

## Development Support

For the entirety of the applications development, the Unity game engine was nominated for its previous interventions with modules already explored by the programme[196]; which was well-adapted for hastening the rate of the games development, as the applications features could be familiarised, as opposed to being learnt. Also, an integrated IDE supported by the Unity engine, Microsoft Visual Studio was nominated for the programmatical additions to the game, due to its previous applications in the programme[197], and for its colour-coded interface; that enabled the behavioural implementations of the game to be addressed quickly. Moreover, as the C# language was also previously taught[198] and integrated within the Unity engine, it was selected as the games fundamental programming language; as well for its OOP disciplines and library support potential.

Complementary to the game's aesthetic representation, Adobe Photoshop was utilised for the creation of the games image assets, given its broad suite of image manipulation tools; and my long experience with operating its interfaces, and creating images for various applications[199]. Accompanying the games graphical elements, Autodesk Maya was employed for the population of the games sophisticated models; this was used as a result of the software's teachings within previous programme modules[200], and for its model export compatibility with the Unity engine.

As for the game's narrative development focuses, Audacity [45] was applicated as a basic audio editing application, for addressing the games ambient, interactive, and transitional sound effects. Also applicated in the other modules of the programme[201], Audacity was nominated for its suite of effects and tools that can be administered to manipulate audio.

# Acknowledgements

For all the staff stationed at De Montfort University, I express my thanks for an unforgettable experience as an undergraduate student and for the facilities, resources, and acceptance onto the programme. I would also like to thank Dr. Jethro Shell as my supervisor and senior lecturer at De Montfort University over the last three years, as an entertainer, lecturer, and role model.

Moreover, I would like to take this opportunity to thank all those who have and are currently supporting the United Kingdom, during the episodic outbreaks of the COVID-19 pandemic.

To my family and friends that have supported me throughout this emotional downfall and heartbreak, during such monumental times, I would like to mention that your efforts will not be overlooked, thank you. Lastly, for Tanniqua Carter, my first love; I have succeeded like I had always planned and made sacrifices for, in ways you will never realize. As a society, we acknowledge the troubles that currently pressure us, but as partners, they are not empathised. I sincerely wish for

[196] Mobile Games I and Mobile Games II.

[197] Mobile Games I, Mobile Games II, Object Orientated Programming in C++, Advanced Object Orientated Programming in C++, Introduction to Shaders, Artificial Intelligence for Simulation, Game Engine Architecture and Game Engine Development.

[198] Within the Mobile Games I and Mobile Games II programme modules.

[199] Graphic design is one of my fundamental hobbies and interest. Ranging use from animated banners to simple logos and avatars.

[200] Within the 3D Modelling module.

[201] Mobile Games I and Mobile Games II.

your return upon this pandemics passing; however, at the time of writing this passage, I am only able to express my gratitude to you for being an impression on my University experience and life. It has been memorable.

You will never be forgotten Sugar, Spice and Lucky, thank you for supporting me through the course of my entry level education; you have been significant to the journey that has led to what I have achieved now. Farewell.

## Bibliography

[1] *Jigsaw*. (2017) [Film] Directed by MICHAEL SPIERIG and PETER SPIERIG. USA: Twisted Pictures.

[2] Fireproof Games (2012) *The Room*. [Online] Mobile. Guildford: Fireproof Games.

[3] PDALIFE.com (2020) *The Room Three*. [Online Image] Available from: https://pdalife.com/the-room-three-android-a18265.html [Accessed: 03/08/20].

[4] Android Police (2013) *The Room*. [Online Image] Available from: https://www.androidpolice.com/2013/04/05/new-game-the-room-ipad-game-of-the-year-2012-is-now-on-the-play-store/ [Accessed: 03/08/20].

[5] Blue Brain Games (2017) *The House of Da Vinci*. [Online] Mobile. Bratislava: Blue Brain Games

[6] Glitch Games (2012) *Forever Lost*. [Online] Mobile. Oxford: Glitch Games.

[7] Droid Gamers (2017) *The House of Da Vinci Flower Tile*. [Online Image] Available from: https://www.droidgamers.com/2017/09/07/the-house-of-da-vinci/ [Accessed: 03/08/20].

[8] Glitch Games (2012) *Forever Lost puzzle one*. [Online Image] Available from: https://glitch.games/portfolio/forever-lost-episode-1/ [Accessed: 03/08/20].

[9] Tencent Games (2019) *Call of Duty: Mobile*. [Online] Mobile. Santa Monica: Activision.

[10] Tencent Games (2017) *PUBG Mobile*. [Online] Mobile. Seoul: PUBG Corporation.

[11] The Indian Express (2020) *PUBG Mobile combat view*. [Online Image] Available from: https://indianexpress.com/article/technology/gaming/pubg-mobile-new-anti-cheat-system-6503832/ [Accessed: 04/08/20].

[12] Gamebraves (2018) *Call of Duty: Mobile multiplayer beta*. [Online Image] Available from: https://www.gamerbraves.com/tencents-call-of-duty-mobile-is-now-in-closed-beta-screenshot-leaked/ [Accessed: 04/08/20].

[13] Haiku Games (2015) *Adventure Escape: Asylum*. [Online] Mobile. Stanford: Haiku Games.

[14] Snapbreak Games AB (2018) *Faraway 3: Artic Escape*. [Online] Mobile. Gothenburg: Snapbreak Games AB.

[15] APK Pure (2016) *Guide Adventure Escape Asylum*. [Online Image] Available from: https://apkpure.com/guide-adventure-escape-asylum/com.cepetkasil.adventureescapeasylum5kamin [Accessed: 04/08/20].

[16] Lelula Games (2019) *Review: Faraway 3 Artic Escape*. [Online Image] Available from: https://girlwithswords.wordpress.com/2019/03/02/review-faraway-3-arctic-escape/ [Accessed: 04/08/20].

[17] *Escape Room*. (2019) [Film] Directed by ADAM ROBITEL. USA: Columbia Pictures.

[18] UNIT9 (2017) *Jigsaw: Escape Room*. [Online Image] Available from: https://www.unit9.com/project/jigsaw-escape-room/ [Accessed: 04/08/20].

[19] Amino (2019) *Escape Room*. [Online Image] Available from: https://aminoapps.com/c/horror/page/blog/escape-room-2019-movie-review/8BwF_mu2NjWZjRxrnjKb28NK87gvgW6 [Accessed: 04/08/20].

[20] Parsec Productions (2012) *Slender: The Eight Pages*. [Online] Available from: http://www.parsecproductions.net/slender/ [Accessed 05/08/20].

[21] Dads Gaming Addiction (2012) *Beta or not, you will be on the edge of your seat*. [Online Image] Available from: https://www.dadsgamingaddiction.com/slender-the-eight-pages/ [Accessed: 05/08/20].

[22] PlayTogether Studio (2019) *Mad Experiments: Escape Room*. [Online] Computer. North America: PlayTogether Studio.

[23] Gameplay Tips (2020) *Silver key object interaction*. [Online Image] Available from: https://gameplay.tips/guides/7557-mad-experiments-escape-room.html [Accessed: 05/08/20].

[24] BALDALF, M. and FROHLICH, P. and ADEGEYE, F. and SUETTE, S. (2015) Investigating On-Screen Gamepad Designs for Smartphone-Controlled Video Games. *ACM Transactions on Multimedia Computing Communications and Applications*. [Online] 12 (22). Available from: https://www.researchgate.net/publication/283283369_Investigating_On-Screen_Gamepad_Designs_for_Smartphone-Controlled_Video_Games [Accessed: 06/08/20].

[25] Cyan Worlds (2017) *realMyst*. [Online] Mobile. South Saskatoon: Noodlecake Studios.

[26] Google Play (2017) *realMyst*. [Online Image] Available from: https://play.google.com/store/apps/details?id=com.noodlecake.realmyst&hl=en_US [Accessed: 06/08/20].

[27] Fireproof Games (2013) *The Room Two*. [Online] Mobile. Guildford: Fireproof Games.

[28] POP, D-PAUL. And A, ALTAR. (2014) Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*. [Online] 69. Available from: https://www.sciencedirect.com/science/article/pii/S187770581400352X [Accessed: 07/08/20].

[29] StackExchange (2012) *MVC Pattern*. [Online Image] Available from: https://softwareengineering.stackexchange.com/questions/136792/is-this-a-proper-implementation-of-an-ios-mvc-pattern [Accessed: 07/08/20].

[30] Unity Technologies (2019) *Unity*. Version 2019.1.6f1 [Software] San Francisco: Unity Technologies.

[31] KUMAR, G. and BHATIA, P. G (2012) Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE).* [Online] 2 (Issue 2). Available from: https://www.researchgate.net/publication/255707851_Impact_of_Agile_Methodology_on_Software_Development_Process [Accessed: 08/08/20].

[32] World Health Organization (2020) *Coronavirus*. [Online] World Health Organization. Available from: https://www.who.int/health-topics/coronavirus#tab=tab_1 [Accessed: 08/08/20].

[33] Autodesk, Inc. (2018) *Autodesk Maya*. Version 2018 [Software] California: Autodesk, Inc.

[34] Unity (2020) *Occlusion culling*. [Online] Unity. Available from: https://docs.unity3d.com/Manual/OcclusionCulling.html [Accessed: 08/08/20].

[35] Unity (2020) *Draw call batching*. [Online] Unity. Available from: https://docs.unity3d.com/Manual/DrawCallBatching.html [Accessed: 08/08/20].

[36] Adobe, Inc. (2017) Adobe Photoshop. Version: CC 2017 [Software] California: Adobe, Inc.

[37] Agile Alliance (2020) *Exploratory Testing*. [Online] Agile Alliance. Available from: https://www.agilealliance.org/glossary/exploratory-testing/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'exploratory*20testing))~searchTerm~'~sort~false~sortDirection~'asc~page~1) [Accessed: 12/08/20].

[38] Sealights (2020) *Understanding Agile Testing Methodology and 4 Agile Testing Methods.* [Online] Sealights. Available from: https://www.sealights.io/agile-testing/understanding-agile-testing-methodology-and-4-agile-testing-methods/ [Accessed: 12/08/20].

[39] NOVOSELTSEVA, N (2019) 8 Benefits of Unit Testing. [Weblog] *DZone*. 30th August. Available from: https://dzone.com/articles/top-8-benefits-of-unit-testing [Accessed: 12/08/20].

[40] RONGALA, R (2015) What is Black Box Testing: Advantages and Disadvantages. [Weblog] *Invensis.* 9th March. Available from: https://www.invensis.net/blog/it/black-box-testing-advantages-disadvantages/ [Accessed: 12/08/20].

[41] TESTBYTES (2019) Black Box Testing Techniques with Examples. [Weblog] *testbytes*. 25th November. Available from: https://www.testbytes.net/blog/black-box-testing/ [Accessed: 12/08/20].

[42] Stackify (2020) *What is code profiling?* [Online] Stackify. Available from: https://stackify.com/what-is-code-profiling/ [Accessed: 12/08/20].

[43] Microsoft Corporation (2019) Microsoft Visual Studio Community. Version: 16.5.5 [Software] Washington: Microsoft Corporation

[44] Stackify (2017) *OPP Concept in C#* [Online] Stackify. Available from: https://stackify.com/oop-concepts-c-sharp/ [Accessed: 12/08/20].

[45] Audacity Team (2015) *Audacity*. Version 2.1.1 [Software] Pennsylvania: Audacity Team

## *Game Assets*

Tudor rose texture. Available from: https://www.freepik.com/free-photos-vectors/carpet-texture [Accessed: 13/08/20].

Old man texture. Available from: https://commons.wikimedia.org/wiki/File:Michiel_Mierevelt_-_Portrait_of_an_Eighty-Year_Old_Man_-_Google_Art_Project.jpg [Accessed: 13/08/20].

Rag man texture. Available from: https://www.oceansbridge.com/shop/artists/r/re-res/rembrandt/portrait-of-an-old-man-15 [Accessed: 13/08/20].

Biblical man texture. Available from: https://fr.m.wikipedia.org/wiki/Fichier:Bust_of_an_old_man_(1631),_by_Rembrandt_van_Rijn.jpg [Accessed: 13/08/20].

Safe dial texture. Available from: https://www.shutterstock.com/image-vector/realistic-combination-safe-lock-isolated-on-465672989?irclickid=23i308RFHxyORZKwUx0Mo3EWUkiTJ9R1nxNLSQ0&irgwc=1&utm_medium=Affiliate&utm_campaign=Ulugbek+Khalilov&utm_source=1425983&utm_term=&c3ch=Affiliate&c3nid=IR-1425983 [Accessed: 13/08/20].

Red book texture. Available from: https://www.wallpaperflare.com/texture-tileable-seamless-book-hard-cover-material-textured-wallpaper-gzelt [Accessed: 13/08/20].

Green book texture. Available from: https://lostandtaken.com/downloads/seamless-book-cover-textures-5/ [Accessed: 13/08/20].

Blue book texture. Available from: https://lostandtaken.com/downloads/seamless-book-cover-textures-7/ [Accessed: 13/08/20].

Crying boy texture. Available from: http://www.theparanormalguide.com/blog/the-curse-of-the-crying-boys [Accessed: 13/08/20].

Blood splatter texture. Available from: https://toppng.com/blood-splatter-PNG-free-PNG-Images_98902 [Accessed: 13/08/20].

Dragon disc texture. Available from: https://www.pngfuel.com/free-png/acyts [Accessed: 13/08/20].

Roof tile texture. Available from: https://architextur.es/textures/roof-tiles / https://www.google.com/search?q=roof+tiles+texture&tbm=isch&ved=2ahUKEwidx4CAkfrpAhUkgM4BHZlcBdMQ2-cCegQIABAA&oq=roof+tiles+texture&gs_lcp=CgNpbWcQAzIECCMQJzICCAAyAggAMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIAFDrCFjrCGC3CmgAcAB4AIABUogBUpIBATGYAQCgAQGqAQtnd3Mtd2l6LWltZw&sclient=img&ei=KlXiXp0KpIC6vg-ZuZWYDQ&bih=938&biw=1920#imgrc=xHtReXh5JJbI7M [Accessed: 13/08/20].

Dragon poster texture. Available from: https://www.pinterest.co.uk/pin/494270127827607890/ [Accessed: 13/08/20].

Chinese cloth texture. Available from: https://www.etsy.com/no-en/listing/263496561/chinese-new-year-decorations-big-fu [Accessed: 13/08/20].

Demonic eyes texture. Available from: https://www.clipartmax.com/middle/m2H7K9Z5K9i8m2m2_eyes-demon/ [Accessed: 13/08/20].

Handprint texture. Available from: https://www.onlygfx.com/5-black-handprints-png-transparent/ [Accessed: 13/08/20].

Book page one texture. Available from: https://www.pngwing.com/en/free-png-kawjc#google_vignette [Accessed: 13/08/20].

Music sheet texture. Available from: http://clipart-library.com/clipart/n1137638.htm

Wall scratch texture. Available from: https://www.nicepng.com/ourpic/u2q8e6i1q8y3o0e6_scratch-marks-png-picture-black-and-white-library/ [Accessed: 13/08/20].

Speaker mesh texture. Available from: https://www.freepik.com/free-photo/speaker-grill-texture-background_1190160.htm [Accessed: 13/08/20].

Girl on rocks texture. Available from: https://www.treksplorer.com/best-travel-backpacks-for-women/ [Accessed: 13/08/20].

Mansion texture. Available from: https://www.pinterest.co.uk/pin/763078730583306478/ [Accessed: 13/08/20].

Lightning texture. Available from: https://www.dreamstime.com/silhouette-lightning-grunge-black-lightnings-over-white-background-image134599478 [Accessed: 13/08/20].

Trees texture. Available from: http://clipart-library.com/clip-art/silhouette-of-forest-4.htm [Accessed: 13/08/20].

Male silhouette texture. Available from: https://www.shutterstock.com/video/clip-3005101-silhouette-man-walking [Accessed: 13/08/20].

Rain texture. Available from: https://www.cleanpng.com/png-rain-icon-floating-rain-395005/preview.html [Accessed: 13/08/20].

Cracked wall normal map. Available from: https://za.pinterest.com/pin/122160208625470109/ [Accessed: 13/08/20].

Squared carpet normal map. Available from: https://www.filterforge.com/filters/13121-normal.html [Accessed: 13/08/20].

Large cracks normal map. Available from: https://www.filterforge.com/filters/9691-normal.html [Accessed: 13/08/20].

Scratched metal normal map. Available from: https://www.filterforge.com/filters/7957-normal.html [Accessed: 13/08/20].

Rusty metal normal map. Available from: https://www.filterforge.com/filters/10523-normal.html [Accessed: 13/08/20].

Canvas paint normal map. Available from: https://everytexture.com/everytexture-com-stock-paint-texture-00033/ [Accessed: 13/08/20].

Seamless grass normal map. Available from: https://www.sketchuptextureclub.com/textures/nature-elements/vegetation/green-grass/clover-grass-texture-seamless-18848 [Accessed: 13/08/20].

Water pattern normal map. Available from: https://www.pinterest.co.uk/pin/627055948096477798/ [Accessed: 13/08/20].

Grass pattern normal map. Available from: https://unrealmethodscom.wordpress.com/2015/11/18/creating-ue4-terrain-with-world-machine-part-iv-final-adding-details-to-the-terrain/ [Accessed: 13/08/20].

Rock pattern normal map. Available from: https://everytexture.com/everytexture-com-stock-rocks-texture-00023/ [Accessed: 13/08/20].

Seamless lava video texture. Available from: https://www.youtube.com/watch?v=o_Xx2cFKMbI [Accessed: 13/08/20].

Crooked man video texture. Available from: https://www.youtube.com/watch?v=aDsUyuE83do [Accessed: 13/08/20].

Dragon model. Available from: https://www.turbosquid.com/FullPreview/Index.cfm/ID/1036048 [Accessed: 13/08/20].

Temple symbol model. Available from: https://www.turbosquid.com/FullPreview/Index.cfm/ID/1054569 [Accessed: 13/08/20].

Piano key model. Available from: https://free3d.com/3d-model/piano-166.html [Accessed: 13/08/20].

Face mask model. Available from: https://free3d.com/3d-model/mask-from-the-film-who-am-i-51027.html [Accessed: 13/08/20].

Microphone model. Available from: https://free3d.com/3d-model/microphone--52251.html [Accessed: 13/08/20].

Sculpture model. Available from: https://free3d.com/3d-model/face-sculpture-on-stick-87277.html [Accessed: 13/08/20].

Desk model. Available from: https://free3d.com/3d-model/-office-desk-v3--821728.html [Accessed: 13/08/20].

Telephone model. Available from: https://www.turbosquid.com/3d-models/3d-landline-telephone-1419155 [Accessed: 13/08/20].

Desk lamp model. Available from: https://free3d.com/3d-model/older-lamp-28459.html [Accessed: 13/08/20].

Clipboard model. Available from: https://www.turbosquid.com/3d-models/free-c4d-model-simple-clipboard/856589 [Accessed: 13/08/20].

Theme song audio. Available from: https://www.youtube.com/watch?v=ziQ9GURNrUg [Accessed: 13/08/20].

Thunder and lightning audio. Available from: https://www.youtube.com/watch?v=10rH37YYvKU [Accessed: 13/08/20].

Button click audio. Available from: https://www.youtube.com/watch?v=O7Gvrug_bko [Accessed: 13/08/20].

Squeaking door audio. Available from: https://www.youtube.com/watch?v=NTds-Wtw3Wg [Accessed: 13/08/20].

Whoosh transition audio. Available from: https://www.youtube.com/watch?v=rvaTKOpRxlQ [Accessed: 13/08/20].

Scene theme audio. Available from: https://www.youtube.com/watch?v=XC3Pdi8K-Cs&t=450s [Accessed: 13/08/20].

Torch click audio. Available from: https://freesound.org/people/dersuperanton/sounds/435845/ [Accessed: 13/08/20].

Light buzz audio. Available from: https://www.youtube.com/watch?v=FYUs2rJAQvo [Accessed: 13/08/20].

Picture frame slide audio. Available from: https://www.youtube.com/watch?v=7cAnlZ1U1Mg [Accessed: 13/08/20].

Picture frame impact audio. Available from: https://www.youtube.com/watch?v=13aOApJL2yY [Accessed: 13/08/20].

Safe door open audio. Available from: https://www.youtube.com/watch?v=yAxYuCCJp7Y [Accessed: 13/08/20].

Safe dial correct number, safe dial rotate clank, safe dial next number audio. Available from: https://www.youtube.com/watch?v=wwV23GKsAu0 [Accessed: 13/08/20].

Safe reset lock audio. Available from: https://freesound.org/people/ingudios/sounds/119494/ [Accessed: 13/08/20].

Safe dial wrong direction audio. Available from: https://www.youtube.com/watch?v=ILtw7SKMjCQ [Accessed: 13/08/20].

Boy screaming audio. Available from: https://www.youtube.com/watch?v=jNk6_4jMHW0 [Accessed: 13/08/20].

Item obtained audio. Available from: https://www.youtube.com/watch?v=ILtw7SKMjCQ [Accessed: 13/08/20].

Place table key audio. Available from: https://www.youtube.com/watch?v=fm3VjiF5Zg8 [Accessed: 13/08/20].

Lava pit audio. Available from: https://www.youtube.com/watch?v=CKUOEvkelhM [Accessed: 13/08/20].

Country marker light audio. Available from: https://freesound.org/people/ascap/sounds/242431/ [Accessed: 13/08/20].

Evil laugh audio. Available from: https://www.youtube.com/watch?v=f3BPx2lt6E0 [Accessed: 13/08/20].

Man screaming audio. Available from: https://www.youtube.com/watch?v=bhr0NWMfl4E [Accessed: 13/08/20].

Draw knife audio. Available from: https://www.youtube.com/watch?v=QEft2eUqKpE [Accessed: 13/08/20].

Knife stab audio. Available from: https://www.youtube.com/watch?v=VOcf3ChmWtQ [Accessed: 13/08/20].

Error buzzer audio. Available from: https://motionarray.com/sound-effects/transition-sound-effects-free-130337 [Accessed: 13/08/20].

Knife pick-up audio. Available from: https://www.youtube.com/watch?v=DBpf5a1L_9E [Accessed: 13/08/20].

Corridor footsteps audio. Available from: https://www.youtube.com/watch?v=spDzJCAL5lk&t=10s [Accessed: 13/08/20].

Door open echo audio. Available from: https://www.youtube.com/watch?v=eXZd094EYus [Accessed: 13/08/20].

Girl grunt audio. Available from: https://www.videvo.net/sound-effect/adult-female-painful-grunt-02/393948/ [Accessed: 13/08/20].

Punch picture impact audio. Available from: https://www.zapsplat.com/sound-effect-category/wood/page/9/ [Accessed: 13/08/20].

Music box picture impact loud, music box picture impact soft audio. Available from: https://www.zapsplat.com/sound-effect-category/wood/page/9/ [Accessed: 13/08/20].

Player jump audio. Available from: https://www.videvo.net/sound-effect/adult-female-exertion-grunt-01/393938/ [Accessed: 13/08/20].

Player traversal audio. Available from: https://www.videvo.net/sound-effect/adult-female-heavy-panting-and-breathing/393941/ [Accessed: 13/08/20].

Player idle inhale audio. Available from: https://www.videvo.net/sound-effect/whoosh-female-breth-sdt032105/261731/ [Accessed: 13/08/20].

Player idle exhale audio. Available from: https://www.videvo.net/sound-effect/whoosh-female-breth-sdt032104/261730/ [Accessed: 13/08/20].

Player footsteps audio. Available from: https://www.videvo.net/sound-effect/footsteps-wood-floor-pe287503/244726/ [Accessed: 13/08/20].

Music box placed impact audio. Available from: https://www.videvo.net/sound-effect/metal-wood-hits-debri-fs021401/251568/ [Accessed: 13/08/20].

Carousel music chime audio. Available from: https://www.youtube.com/watch?v=YaeHtp0Pp4o [Accessed: 13/08/20].

Carousel button pressed audio. Available from: https://www.videvo.net/sound-effect/switch-on-antique-button-pehd086303/258134/ [Accessed: 13/08/20].

Carousel clacking audio. Available from: https://www.videvo.net/sound-effect/floppy-drive-spinnin-pe412301/244371/ [Accessed: 13/08/20].

Carousel shaking audio. Available from: https://www.videvo.net/sound-effect/door-rattle-05/416954/ [Accessed: 13/08/20].

Music box chime audio. Available from: https://www.youtube.com/watch?v=u9WsZoceais [Accessed: 13/08/20].

Handprint scare audio. Available from: https://www.youtube.com/watch?v=TTQ8Oh_aBiA&list=PLcSgqX6WCxoh9eIZ5bNIWTNAQlTb8tH7C&index=8 [Accessed: 13/08/20].

Gate creak open audio. Available from: https://www.youtube.com/watch?v=AQ1SE4tLICg&list=PL634EA6C284405875&index=74 [Accessed: 13/08/20].

Chase theme audio. Available from: https://www.youtube.com/watch?v=nEuXiJ-d2YM [Accessed: 13/08/20].

Evil man grunt one audio. Available from: https://www.videvo.net/sound-effect/human-grunt-25/428355/ [Accessed: 13/08/20].

Evil man grunt two audio. Available from: https://www.videvo.net/sound-effect/human-grunt-39/428369/ [Accessed: 13/08/20].

Evil man roaming chime audio. Available from: https://www.youtube.com/watch?v=zclD93VnpEw [Accessed: 13/08/20].

Evil man grunting audio. Available from: https://www.videvo.net/sound-effect/male-breaths-labored-az174802/250568/ [Accessed: 13/08/20].

Piano lid creak audio. Available from: https://www.videvo.net/sound-effect/creak-hinge-01/413534/ [Accessed: 13/08/20].

Piano lid impact audio. Available from: https://www.videvo.net/sound-effect/piano-hit/435856/ [Accessed: 13/08/20].

Piano key scrape audio. Available from: https://www.videvo.net/sound-effect/piano-scrape-07/435868/ [Accessed: 13/08/20].

Wardrobe knocking audio. Available from: https://www.videvo.net/sound-effect/knocking-on-wood-door-fist-pehd097902/249148/ [Accessed: 13/08/20].

Wardrobe door squeak audio. Available from: https://www.videvo.net/sound-effect/door-wood-69/417204/ [Accessed: 13/08/20].

Insert wind-up key audio. Available from: https://www.videvo.net/sound-effect/switch-breaker-03/442890/ [Accessed: 13/08/20].

Insert latch key audio. Available from: https://www.videvo.net/sound-effect/switch-click-vo5850-pe1090808/258076/ [Accessed: 13/08/20].

Latch key unlock audio. Available from: https://www.videvo.net/sound-effect/35mm-camera-click-wi-pe1012702/233828/ [Accessed: 13/08/20].

Lid hinge propped open audio. Available from: https://www.videvo.net/sound-effect/switch-small-toggle-04/442849/ [Accessed: 13/08/20].

Place disc audio. Available from: https://www.videvo.net/sound-effect/metal-impacts-dull-fs010301/251285/ [Accessed: 13/08/20].

Align disc needle audio. Available from: https://www.videvo.net/sound-effect/latch-swing-close-squeak-door-secur-pehd047102/249424/ [Accessed: 13/08/20].

Light switch on audio. Available from: https://www.videvo.net/sound-effect/light-switches-click-pe803705/249818/ [Accessed: 13/08/20].

Light switch off audio. Available from: https://www.videvo.net/sound-effect/light-switches-click-pe803703/249816/ [Accessed: 13/08/20].

Place book audio. Available from: https://www.soundsnap.com/search/audio/wood%20impact/score [Accessed: 13/08/20].

Turn page audio. Available from: https://www.videvo.net/sound-effect/book-large-23/406530/ [Accessed: 13/08/20].

Book cover impact audio. Available from: https://www.videvo.net/sound-effect/book-hardcover-01/406619/ [Accessed: 13/08/20].

Strobe light flash audio. Available from: https://www.videvo.net/sound-effect/camera-flash-pot-go-crt2014603/238304/ [Accessed: 13/08/20].

Paper impact audio. Available from: https://www.videvo.net/sound-effect/target-hit-light-pap-pe1108506/258688/ [Accessed: 13/08/20].

Piano chime audio. Available from: https://www.youtube.com/watch?v=XUXsKOkhoFM [Accessed: 13/08/20].

Door unlock audio. Available from: https://www.videvo.net/sound-effect/wood-chair-scrape-pe1013803/262133/ [Accessed: 13/08/20].

Telephone ringing audio. Available from: https://www.videvo.net/sound-effect/phone-ring-digital-am-ee114901/253237/ [Accessed: 13/08/20].

Telephone error tone audio. Available from: https://www.videvo.net/sound-effect/telephone-hang-up-dial-tone-l-pe598704/258946/ [Accessed: 13/08/20].

Telephone pick-up, telephone hang-up audio. Available from: https://www.videvo.net/sound-effect/telephone-p-u-hang-up-rotary-p-pe596204/258979/ [Accessed: 13/08/20].

Glass break audio. Available from: https://www.videvo.net/sound-effect/glass-small-break-pe1051017/245631/ [Accessed: 13/08/20].

Telephone outbound call audio. Available from: https://www.videvo.net/sound-effect/telephone-ring-tone-pe598801/258991/ [Accessed: 13/08/20].

Women struggling audio. Available from: https://www.videvo.net/sound-effect/adult-female-struggling-grunts/393981/ [Accessed: 13/08/20].

Telephone dropped audio. Available from: https://www.videvo.net/sound-effect/telephone-drop-01/444731/ [Accessed: 13/08/20].

Evil man laughing telephone audio. Available from: https://www.videvo.net/sound-effect/laugh-evil-old-man-pe974801/249434/ [Accessed: 13/08/20].

Telephone suspense audio. Available from: https://www.youtube.com/watch?v=0vKMa7-yAQg [Accessed: 13/08/20].

Telephone disconnected tone audio. Available from: https://www.videvo.net/sound-effect/brit-telephone-number-unobt-pe594501/237375/ [Accessed: 13/08/20].

# Appendices

***Appendix A:***

# Literature Review

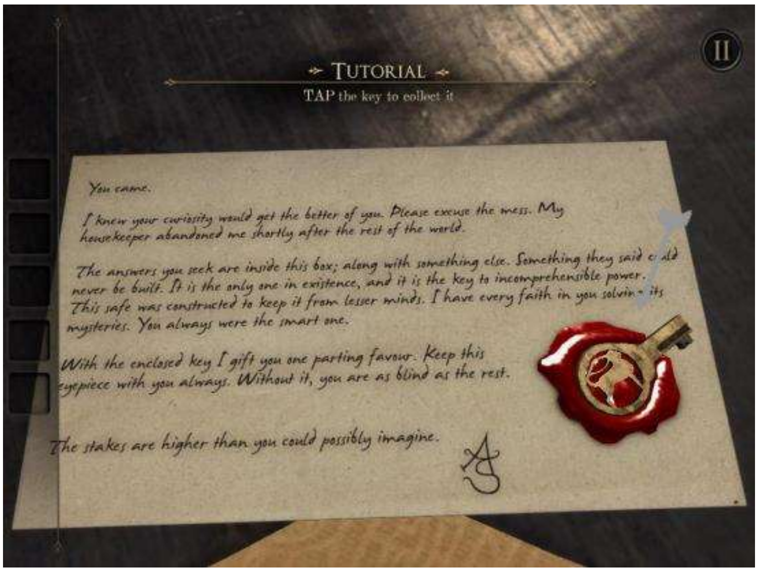
## Introduction

In relation to my study focus of three-dimensional puzzler games for mobile platforms, I am conducting a series of literature reviews to better inform my decisions for developing my proposed game; from which, I aim to gain an understanding for what is considered similar and different within existing games across the genre. Similarities between games are often referred to as “genre conventions”, whereby, games bound by the same genre typically focus upon “mechanics and game design patterns that deliver a particular play experience” [1]. Incorporating genre conventions that can be identified across multiple games, would purpose for my game to have an establishment within the puzzle game genre and therefore provide a sense of “familiarity” [1] for players. Meanwhile, implementing techniques that are not considered generic or common, would enable my game to be considered unique and innovative; this can be achieved by identifying game design similarities and differentiating from them, examples being the “theme and game objective”. These game design components are considered “separate from genre” according to Tulia-Maria [1]. Providing a sense of differentiation could also increase the level of game immersion for players, in which, players may find “events in the story” of the game less ‘predictable’ [1]. In acknowledgement of these research objectives, the literature reviews will make attempt to provide focus upon the puzzle game genre, as my reporting and development focus. To note, there is a lack of academic-based literature available, to accommodate for this research focus entirely.

Throughout the reviews of each literature piece, I will specifically explore the various aspects of mobile puzzler games in respect of aesthetics, mechanics, narrative, user interface (UI) and level design; for which, I believe to be the fundamental game design concepts. Existing games will be explored to discuss these concepts mostly, for which, my methodology is to “find the common components in the games that are used to exemplify the genre” [2]. Moreover, I also intend to explore films to help navigate the narrative, level design and aesthetic components of my game; whereby, the proposed theme of the game aims to resemble a sinister and mystery type setting throughout. My intentions for the theme, are to encourage suspenseful and slowed gameplay to inspire players to interact with their environments cognitively; for such, techniques used within the thriller and mystery genres of film, will be explored to reinforce the theme I am intending to create. As the theme of my game focuses predominantly upon player emotion, I will further investigate the effects that such games pose, on player emotions.

## Reviewing literature

As my initial source of interest, I had wanted to explore the design aspects of video games bound by the puzzle genre. As referred to previously, understanding the design aspects the ‘genre exemplifies’ [2], enables the identification of “genre conventions” [1]; for my game to be recognised as a production of the genre, it requires to adopt some of the techniques recurring in existing puzzle games. For which, the article “The study of Principles of Puzzle Game Design” [3] explores the principles of puzzle game design in focus of “graphics”, “sounds”, “interaction and feedback” as well as “storylines and gameplay”. Regarding “graphics”, the article explores the application of images within puzzle games to take into consideration of the “size of the images” [3], in relation to player restraints on the ability to see images clearly and to see other world spaces; the scaling of images is thereby considered to be a significant design concept. The mobile puzzle game series ‘The Room’ [4], attempts to visualise environments scaled to the real-world and makes use of images that are centric to puzzle resolution; the series achieves image clarity from the use of the movement mechanics of a single camera, that can be controlled to increase the focus level on image detailing within the game world.



You came.

I knew your curiosity would get the better of you. Please excuse the mess. My housekeeper abandoned me shortly after the rest of the world.

The answers you seek are inside this box; along with something else. Something they said could never be built. It is the only one in existence, and it is the key to incomprehensible power. This safe was constructed to keep it from lesser minds. I have every faith in you solving its mysteries. You always were the smart one.

With the enclosed key I gift you one parting favour. Keep this eyepiece with you always. Without it, you are as blind as the rest.

The stakes are higher than you could possibly imagine.

AS

*Image 1: The Room, camera focus upon a letter for image readability [5]*

Moreover, in relation to audio elements of puzzle game design, the authors describe the application of “sounds or background music” to “match the games” [3] and expands to mention the requirement of audio to be suited to all players. Within the context of my proposed game, audio will be suited to allow progression in gameplay for players in any culture or environment; multicultural agility has to be addressed, therefore linguistic audio will be excluded. This technique is also applicated within The Room series [4] also, whereby the prominence of sound is exemplified in the forms of backgrounding music, as well as interactive and transitional sound effects [6]. Furthermore, in focus of player interaction and feedback, the article describes the “most important principle” to be the friendliness of the games interface [3]; this takes into consideration the clarity of the games objectives, the order of which the objectives are presented in ‘logically’, the appearance of the interface, the ease of understanding game mechanics and the possibility of players being able to determine the pace of gameplay. In essence, the authors refer to the simplification of game interaction, for a more comprehensive player experience. In relation to my proposed game, as a mobile-based game, interaction intends to be communicated through the means of buttons, as well as screen tap orientated input on game world objects; buttons will collectively form the user interface for the game. From the implementation of said game interaction, dictates a range of easy to understand game mechanics for players. Meanwhile, the order in which puzzles can be solved, intends to be sequential and portray linearity; this is to maintain player cognition and avoid deterring the player from any fabricated frustration. However, as previously mentioned, the game theme intends to control the players pace of gameplay, as opposed to players controlling the “pace of the game” [3]; in which, players will be encouraged to play slowly. Within ‘The House of Da Vinci’ series, game interactions are considered to be simple from the requirement of finger swipe or tap gestures and are enhanced by the use of recurring mechanics also [7]. An exampling game interaction that satisfies this claim, is the rotary mechanism of objects; which is typically controlled by circular finger swipe motions [8]. Moreover, the appearance of the game’s user interface can also be considered simple; this is due to the lack of buttons that enables it to be easily navigable.

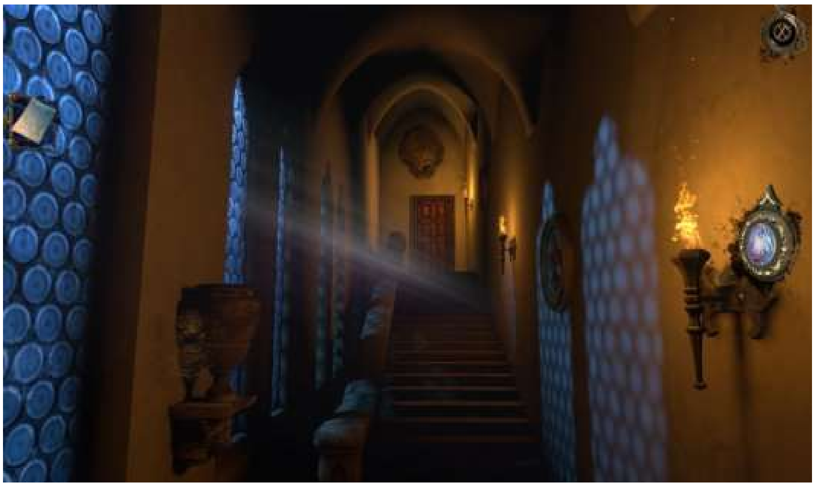
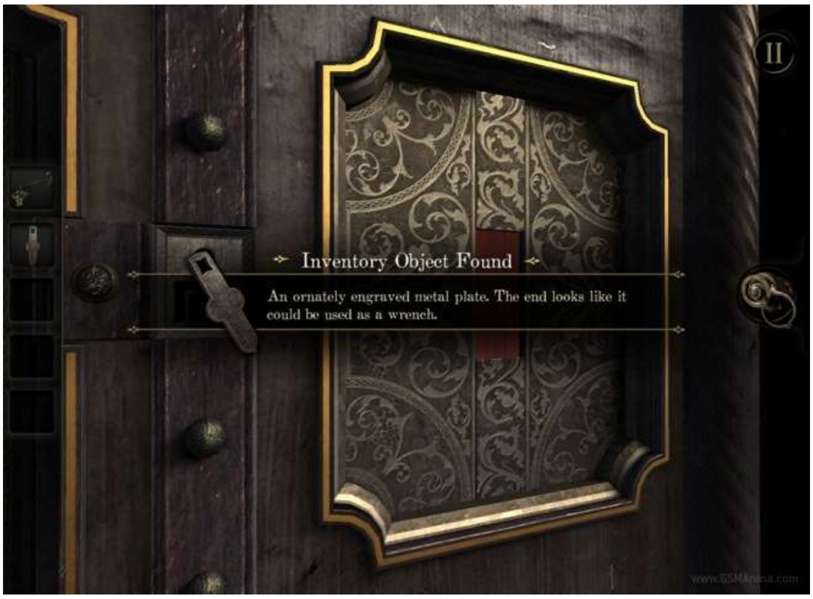


*Image 2: The House of Da Vinci, showcasing the games graphical user interface [9]*

Lastly, in focus of the narrative element of puzzle game design, the article presents "storylines and gameplay" to "offer information for the players", such information is presented as the "main purpose" of puzzle games [3], as the authors describe. In which, the concept of 'storylines' is discussed further as the 'creator of game situations'; which allows for "goals and challenges" to be implemented and for players to attempt and "achieve" them [3]. Meanwhile, the authors refer to "gameplay" as the process in which players 'interact' with the puzzle components within games and become 'entertained' from doing so; the article concludes the purpose of puzzle games, for the production of "challenges and interesting interactions" [3]. In further relation to both The Room series [4] and The House of Da Vinci [7] series of games, the "storylines" introduce "goals and challenges" through the occurrences of cutscenes [10]; these are typically queued, when players overcome the preceding "challenges" [3] in a single area of the game world, or are simply introduced to a new environment or story event. Cutscenes purpose for the transition between "game situations", which allows for newer "goals and challenges" [3] to be presented to players, linearly. Meanwhile, the concept of 'information offering' is typically provided through the means of letters [5] throughout the game world, as well as prompting players with clues.



Inventory Object Found

An ornately engraved metal plate. The end looks like it could be used as a wrench.

*Image 3: The Room, player prompted with a clue for the use of a found object [11]*

Moreover, the “gameplay” element [3] across the games is presented through the interaction with static and obtainable objects; individually, each object represents to be a component of a puzzle, but when the objects are interacted with in a specific sequence, players can achieve the “goal” [3], being puzzle resolution.

Concluding my findings of the article, it is inevitable that the “graphics”, “sounds”, “interaction and feedback” as well as “storylines and gameplay” [3], are significant puzzle game design aspects that need to be considered when producing my own game. It is further apparent that the intricacies of images and audio, can determine whether a player can or cannot progress through a game. As well, the order of game events poses a significance for player cognition; narrative and interaction within my game, therefore requires some form of linearity in the sequence of presenting game events. However, my findings of puzzle game design aspects are conceptual, in which, the authors discuss puzzle game design superficially; the lack of exampling techniques that each of the concepts employ, is a limitation of this paper, hence the need for exemplifying said concepts in existing puzzle games.

For my second source of interest, I wanted to investigate the affect that game elements have upon player emotions; as previously mentioned, suspense is the focal emotion I am wanting to engage throughout the theme of the game. For which, the article “Optimizing Player and Viewer Amusement in Suspense Video Games” explores suspense, as an emotion that can be achieved in games through signalling “the location of a threat” to players; this concerns the amount of “information provided to the audience” [12], to achieve such. In relation to suspense, the authors describe “suspenseful narratives” to typically adopt “first or third-person” camera perspectives, so that players have “the same visual and audible information as the main character” [12]; this technique can be seen within the horror game ‘Slender: The Eight Pages’ [13].

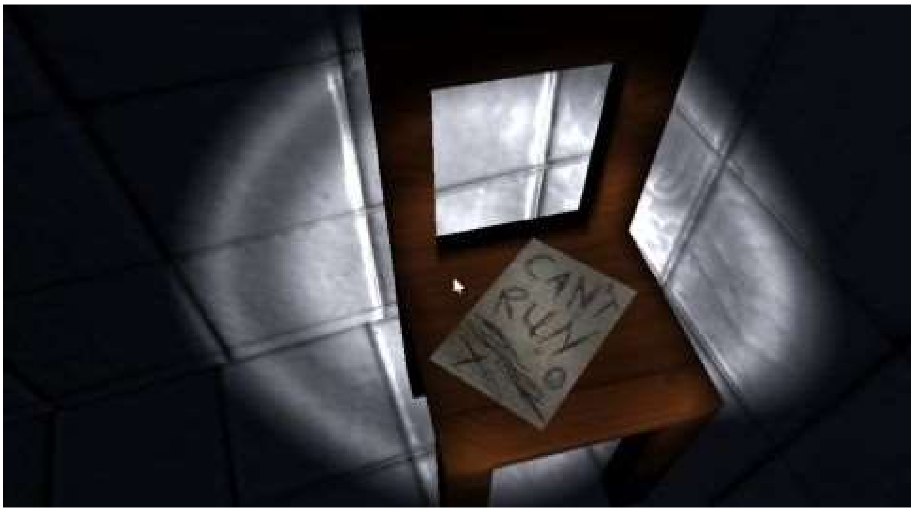


*Image 4: Slender: The Eight Pages, first-person camera perspective [14]*

Moreover, the article continues to explore suspense simulation in video games, in similarity of film productions. In which, the article discusses player experience to involve aspects of “exploration”, "traps”, “persecution” and “claustrophobic environments” [12]. All of which aspects of player experience, reside within Slender[13] and both The Room [4] and The House of Da Vinci [7] game series’.

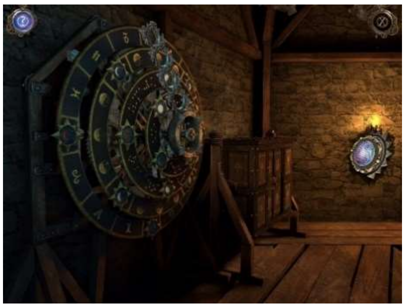


Image 5: The House of Da Vinci, environment [15]

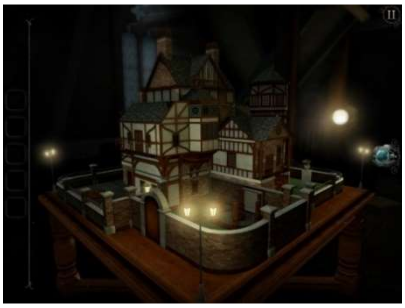


Image 6: The Room: Old Sins, environment [16]

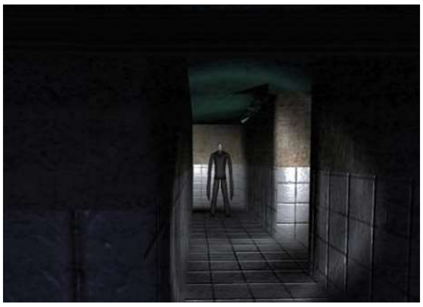


Image 7: Slender: The Eight Pages, environment [17]

In further scope, a film which exemplifies all of these aspects is ‘Jigsaw’ [18], which presents the narrative of people imprisoned within “claustrophobic environments”, to escape, they are required to ‘explore’ and solve a series of puzzles in the forms of “traps” [12]. Furthermore, the authors describe “suspenseful situations” to be communicated to players, via the application of “visual images, text, music, speech and environmental effects”; of which, sounds are said to be better suited to signally a “situation” before a player is able to “see it” [12]. The article continues to comment that ‘changing sounds’ or ‘visually modifying the environment’, ‘are strategies typically used in suspenseful video games’ [12]. In which case, many of the exemplified games [4] [7] [13] and films [18] [19] adopt low-key lit environments and dark colour pallets to heighten “frightening cues that increase the emotional response, without the need of changing the viewport” [12]; this is a recurring concept I want to implement within my production. Moreover, relating to sounds, sounds typically change in dynamic throughout films [18] [19] and games [4] [7] [13], the authors claim that “finding the right moment to provide the information” is essential for developing players “expected emotional response” [12]. Furthermore, information provided to players is considered conversely in respect of players becoming “stressed” when “a lack of information” is given; this infers increasing player challenge but increase in “level of suspense” also [12].

In summary of my findings, the authors have informed the ways in which suspense is orchestrated within video games and the techniques that can be used to fabricate an “emotional response” [12] from players. Additionally, the article and exemplifying sources have explored game design choices in regard to level design, narrative development, and aesthetics; in which, discussions regarding “suspenseful narratives” [12], has mostly informed me with considerations for my own game. Although the articles focus is predominantly upon the horror game genre, the authors do not concern the interactive mechanics of horror games; but alternatively, the aspects of game “theme and game objective” [1], this thereby fulfils its usefulness for this investigation and poses no restraints on relevance to my own production.

## References

[1] CASVEAN, T.P. (2015) An Introduction to Videogame Genre Theory. Understanding Videogame Genre Framework. *Athens Journal of Mass Media and Communications.* [Online] Vol. 2. (Issue 1). Available from: https://www.semanticscholar.org/paper/An-Introduction-to-Videogame-Genre-Theory.-Genre-C%C4%83%C8%99vean/4dd13db214c63cd19c9d0e97df098170dd75dbc0 [Accessed 17/12/19].

[2] BJORK, S. and HOLOPAINEN, J. and LUNDGREN, S. (2003) Game Design Patterns. In: *Digital Games Research Conference, The Netherlands, November 2003.* Rockland: Charles River Media, p. 4.

[3] ZHOU, Z. and WU, L. (2012) The Study of Principles of Puzzle Game Design. In: *2012 International Symposium on Information Technologies in Medicine and Education, Japan, August 2012*. Hokkaido: IEEE, pp. 1-2.

[4] Fireproof Games (2012) *The Room*. [Online] Mobile. Guildford: Fireproof Games.

[5] GameZebo (2013) *The Room Walkthrough*. [Online] Available from: https://www.gamezebo.com/2013/09/03/room-walkthrough-cheats-strategy-guide/ [Accessed 17/12/19].

[6] Techzamazing (2015) *The Room Three (iOS/Android) Gameplay Walkthrough - Part 1*. [Online film] Available from: https://www.youtube.com/watch?v=XFl93xkH75M [Accessed 17/12/19].

[7] Blue Brain Games (2017) *The House of Da Vinci*. [Online] Mobile. Bratislava: Blue Brain Games.

[8] Techzamazing (2017) *The House Of Da Vinci - Walkthrough Gameplay (iOS / Android / STEAM )- PART 1*. [Online film] Available from: https://www.youtube.com/watch?v=KAYk8F3wo5E&t=538s [Accessed 17/12/19].

[9] Adventure Gamers (2018) *The House of Da Vinci*. [Online] Available from: https://adventuregamers.com/articles/view/34525 [Accessed 17/12/19].

[10] Techzamazing (2019) *The House of Da Vinci 2 Gameplay Walkthrough (Android, iOS, Steam) - Part 1*. [Online film] Available from: https://www.youtube.com/watch?v=YOG1pJ5okXI [Accessed 18/12/19].

[11] PRASAD (2012) 'The Room' for iPad game review. [Weblog] *BLOG.GSMARENA*. 7th October. Available from: http://blog.gsmarena.com/the-room-for-ipad-game-review/ [Accessed 18/12/19].

[12] DELATORRE, P. and LEON, C. and HIDALGO, A.S. and TAPSCOTT, A. (2019) Optimizing Player and Viewer Amusement in Suspense Video Games. *IEEE Access*. [Online] Vol. 7. Available from: https://ieeexplore.ieee.org/document/8742555 [Accessed 18/12/19].

[13] Parsec Productions (2012) *Slender: The Eight Pages*. [Online] Available from: http://www.parsecproductions.net/slender/ [Accessed 18/12/19].

[14] WYCISLIK-WILSON, M. (2017) Slender: The Eight Pages review. [Weblog] *techradar*. 27th July. Available from: https://www.techradar.com/uk/reviews/pc-mac/software/slender-the-eight-pages-review-1326593/review [Accessed 18/12/19].

[15] AppUnwrapper (2017) 'The House of Da Vinci' Review: This Seems Familiar. [Weblog] *AppUnwrapper*. 25th June. Available from: https://www.appunwrapper.com/2017/06/25/the-house-of-da-vinci-review/ [Accessed 18/12/19].

[16] REEVES, B. (2018) The Room: Old Sins. [Weblog] *gameinformer*. 23rd January. Available from: https://www.gameinformer.com/games/the_room_old_sins/b/ios/archive/2018/01/23/the-room-old-sins-game-informer-review.aspx [Accessed 18/12/19].

[17] GAME FRONT (2012) Game Front Primer: Everything You Need for Slender: The Eight Pages. [Online] Available from: https://www.gamefront.com/games/gamingtoday/article/game-front-primer-everything-you-need-for-slender-the-eight-pages [Accessed 18/12/19].

[18] *Jigsaw*. (2017) [Film] Directed by MICHAEL SPIERIG and PETER SPIERIG. USA: Twisted Pictures.

[19] *Insidious: Chapter 2*. (2013) [Film] Directed by JAMES WAN. USA: Blumhouse Productions

***Appendix B:***

# Software functional requirements

Functional requirements of software applications represent the requirements of software component functionality, behaviourally. Outlining functional requirements of software is necessary for ensuring that users of the software can accomplish tasks, in use of it. In context of video games and most of all my production focus, a puzzle game, there can only be one user type, a player. The requirements of the software therefore purpose to enable a player to complete all of the set objectives throughout the game; this enables players to reach their goal. This document aims to identify the functional requirements of my production and provide insight into the way in which players can use the software.

## Game overview

'Under Lock and Key' proposes to be a three-dimensional puzzle game, that can be viewed from the first-person and third-person perspectives. The basis of the game pursues the narrative of the playing character, who becomes captured and then imprisoned within a house-type environment; inside each room accessible to the player, features a series of interconnected puzzle components. Players are required to explore and interact with their environments, to solve all of the puzzles within every accessible room. For every room that is completed by the player, nears them to escaping the prison. Each room represents a level to note.

## Basic functional requirements of the game

The basic functional requirements of the software application revolves around a player's ability to interact with menus and initial scenes, which enables a game to be in a state of play or configuration. The following content lists these requirements.

**Functional requirements**

- The application will allow a user to be a player
- The application will allow a player to start a game session
- The application will allow a player to pause the state of a game session
- The application will allow a player to toggle the active state of subtitling
- The application will allow a player to exit a game session
- The application will allow a player to transition between the game's scenes

## Functional requirements of game scenes

The functional requirements of the game's scenes, purposes to outline the ways in which scenes are designed to enable a player to interact with objects within their environments and complete objectives. For which further allows players to approach the end goal of the game: escape. The following content lists these requirements.

**Functional requirements**

- The scenes will allow the presence of interactable game objects
- The scenes will allow the presence of non-interactable game objects
- The scenes will allow the presence of plains to surface the game objects
- The scenes will allow the presence of a player-controlled game object
- The scenes will allow the presence of numerous player-controlled camera objects
- The scenes will allow the presence of game physics, provided by the existence of game object collider and rigid body components
- The scenes will each represent a level, excluding the menu and initial game scenes
- The scene will allow the presence of non-linguistic audio, in the forms of instrumental music and sound effects
- The scenes will allow the presence of light objects, to ensure the visibility of each scene
- The scenes will allow the presence of user interface objects, to enable the movement and interaction of player-controlled and interactable game objects
- The scenes will allow the presence of text overlaying, to linguistically represent sound effects in the form of subtitles

## Functional requirements for the player of the game

The functional requirements of the game in relation to the player, determines how a player is able to control and introduce an interaction with objects within their environments, with the aims of attempting to complete the games objectives. The following content lists these requirements.

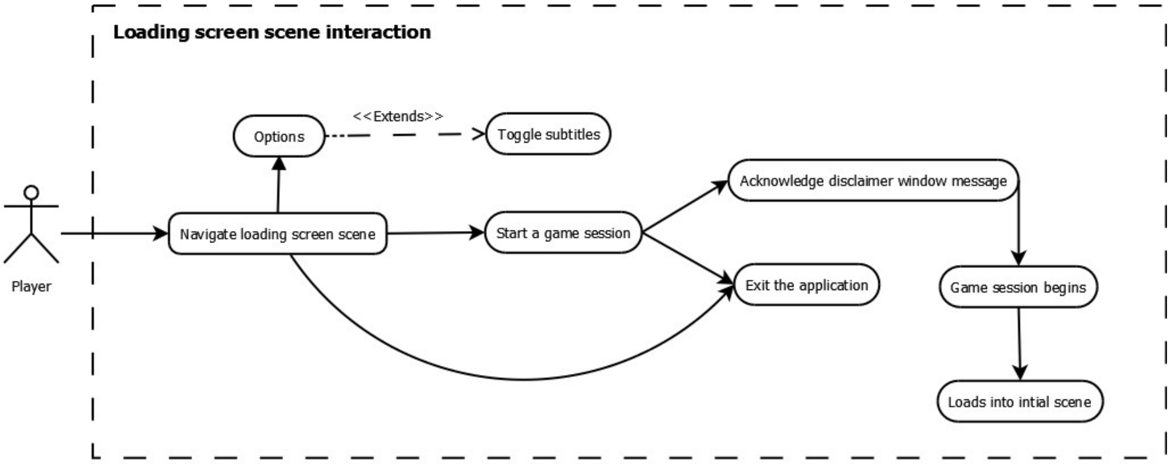
**Functional requirements**

- The player-controlled game object can traverse in the facing direction of the active camera object, when the corresponding user interface buttons are interacted with
- The player-controlled game objects facing direction can be adjusted, when the corresponding user interface buttons are interacted with
- The player-controlled game object can interact with interactable objects, when within the required proximity of these objects, through touch, hold and swipe interactions
- The player-controlled game object should remain upright, in the event of colliding with other game objects
- The player-controlled game object should not be able to leave the intended area of play, unless all of the events within the current level are recognised as being complete (puzzle logic)

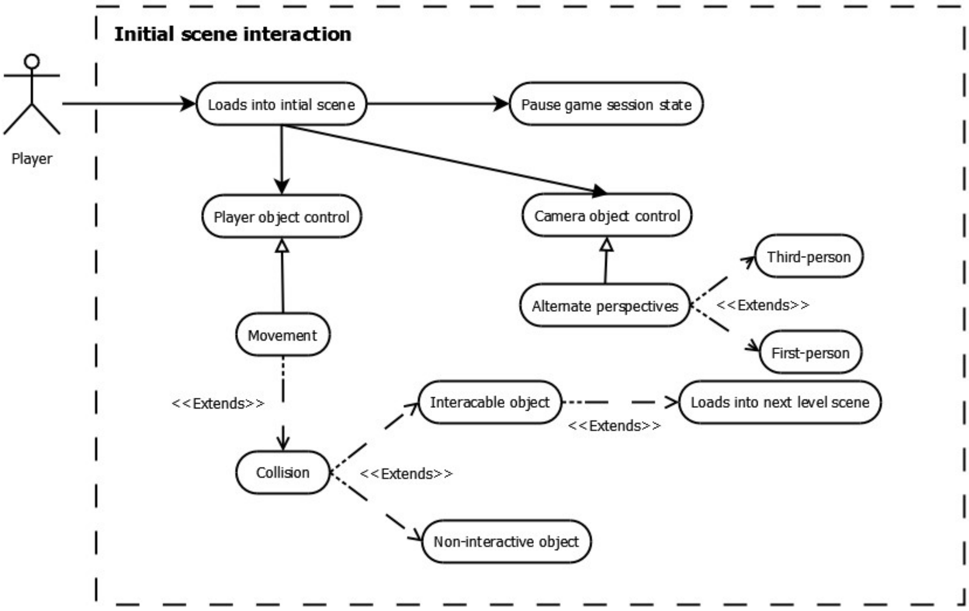
# Software use cases

Software use cases exist for describing the relation between the way software applications are interacted with, by external users. The focus of software use cases is to illustrate the ways in which users can achieve particular goals and the requirements of the software to enable such. In context of my production, the following section of content will outline player relationships with my game proposal, in attempt of showcasing the software's functional requirements, as specified above.

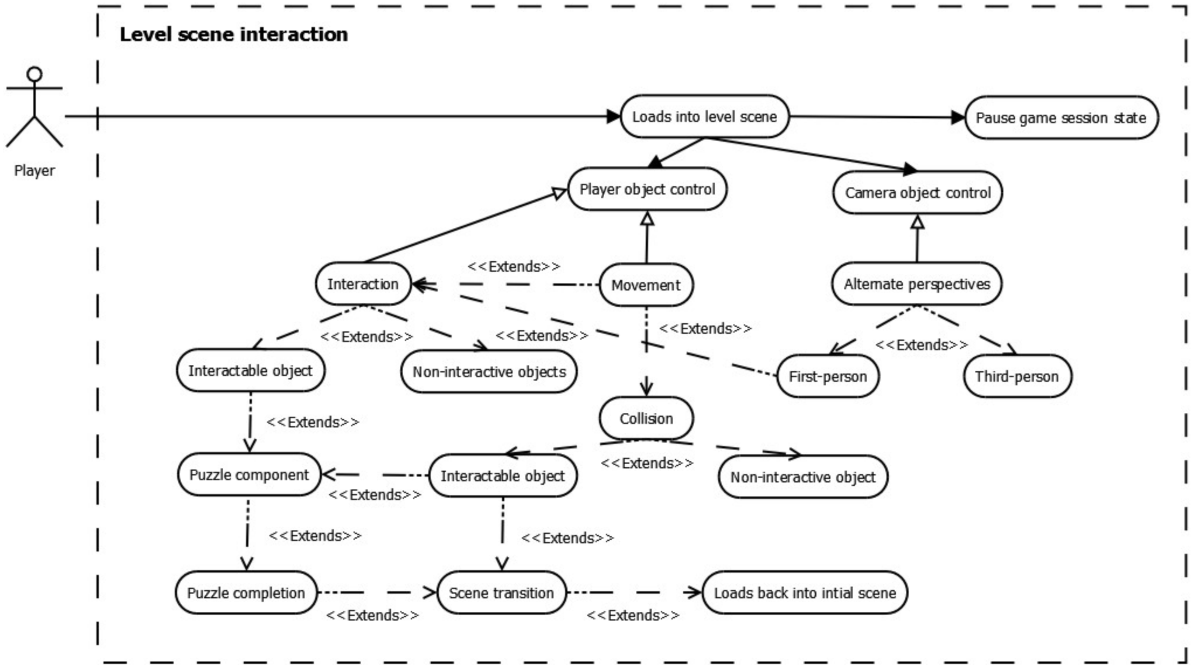
**User case:** loading screen scene interaction



**User case:** initial scene interaction



**User case:** level scene interaction



***Appendix C:***

# Methodology of testing

The purpose for testing, suits the requirement of acknowledging proper functionality within software applications. In the scope of my project proposal, a mobile based game, a series of test cases will be conducted to ensure that the software meets its technical and functional expectations. Thereby, the purpose of this document is to describe and illustrate the testing strategy of the software outlined; the document will provide indicative test cases upon the initial software setup, to provide insight into how test scenarios are orchestrated and therefore measured.

## Test objectives

In relation to testing objectives, the mobile game will be tested in validation of gameplay quality, as well as application reliability, usability, and performance. For which, testing will conform to unit, Blackbox and performance profiling, testing approaches; these methods are suited for testing both a software's internal and external design. As previously mentioned, the software testing will be tailored to the games technical and functional requirements.

In relation to quality testing of the software, the game will be tested in accordance with its expectations of being bug-free, to enable the best gameplay experience for players possible. The nature of these tests will predominantly be focused upon the mechanism components of the game. Meanwhile, relating to application reliability, usability and performance, the game will be tested in relevance of performance profiling; this serves to provide players with a stable and therefore enjoyable gameplay experience. This provides emphasis on the management of graphical representation and vastness of level design for the game.

## Test strategy

Regarding the development of my software piece, I have chosen to adopt the Agile methodology of software development, SCRUM, to enable constant and gradual inspection and adaptation to the software, in the form of sprints. This allows for me to remain comprehensive about the application I am developing and aims to increase the quality of my software deliverable.

In correspondence to testing, I am similarly going to adopt an Agile software testing methodology; the purpose of this methodology is to align the testing processes with the development processes of the software. In which, I propose that the game is tested compositionally, after each component is implemented within the game; this enables the code base of the game to be more robust and bug-free, as the games development becomes increasingly expansive. Moreover, in respect of future development, as defective functionality can be detected and removed gradually, the code base of the software will be enabled to be simplified and easier to interpret. I believe that this strategy of testing will be better suited for locating functionality defects and performance consumption issues, progressively; in which, the most recent implementation would indicate to be the most probable cause for error.

In relation to the testing methodology, I have selected exploratory testing as my testing basis. For which, exploratory can be used for testing the expected behaviours and functionality of software components, throughout its development. Test cases are not typically configured in advance of developing software functionality and are instead designed and executed simultaneously. This allows for the code base of software to be refactored, to enable aspects of the software to behave as expected, which can be determined from the pass rate of the test cases produced. It can also be considered beneficial for discovering component errors, that other testing techniques may ignore.

## Blackbox testing

### Purpose of Blackbox testing

Blackbox testing is a software testing technique which focuses on application behaviour and performance; the code base of a given application is not known and so the functionality of a given application is determined to be working correctly, incorrectly or not working from a visual basis. This is measured from differences observed in application output, from calling various inputs in the application, differences in application output are subject to the working functionality of the inputs to note.

In relation to the mobile game as a piece of software, Blackbox testing enables the identification of lacking and non-functioning game mechanics at a surface level; as there is no need to review the games code base to acknowledge the implemented functionality's working status. Being able to construct test cases quickly aids the pace of development, in which errors can be quickly acknowledged also; therefore, I have nominated this testing method for its suitability.

### Blackbox testing cases

| Case | Summary | Process | Actual result(s) | Expected result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Player object travels upwards and descends gradually when jump button is pressed. | Jump button embedded within the UI is pressed when the player object is 'grounded'. | Player object travels upwards once jump button is pressed; the player object gradually descends due to gravity parameter set. | Player object travels upwards once jump button is pressed; the player object gradually descends due to gravity parameter set. | 🟩 |
| 2 | Player object rotates around its own axis positively in the 'Y' axis when turn right button is pressed or held. | Turn right button embedded within the UI is pressed. | Player object rotates right around its own axis in the 'Y' axis at a constant rate once turn right button is pressed or held. | Player object rotates right around its own axis in the 'Y' axis at a constant rate once turn right button is pressed or held. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 3 | Player object rotates around its own axis negatively in the 'Y' axis when turn left button is pressed or held. | Turn left button embedded within the UI is pressed. | Player object rotates left around its own axis in the 'Y' axis at a constant rate once turn left button is pressed or held. | Player object rotates left around its own axis in the 'Y' axis at a constant rate once turn left button is pressed or held. | 🟩 |
| 4 | Player object travels forwards to its current facing direction when move forward button is pressed or held. | Move forward button embedded within the UI is pressed. | Player object travels forwards to its current facing direction at a constant rate when the forward button is pressed or held. | Player object travels forwards to its current facing direction at a constant rate when the forward button is pressed or held. | 🟩 |
| 5 | Player object travels backwards from its current facing direction when move backward button is pressed or held. | Move backward button embedded within the UI is pressed. | Player object travels backwards from its current facing direction at a constant rate when the backward button is pressed or held. | Player object travels backwards from its current facing direction at a constant rate when the backward button is pressed or held. | 🟩 |
| 6 | Camera perspective changes when camera cycle button is pressed. | Camera cycle button embedded within the UI is pressed. | Camera perspective changes, via disabling the current camera at the time the camera cycle button is pressed and enabling the disabled camera at the time the camera cycle button is pressed. | Camera perspective changes, via disabling the current camera at the time the camera cycle button is pressed and enabling the disabled camera at the time the camera cycle button is pressed. | 🟩 |
| 7 | Player object deflects off objects, does not glitch through other rigid objects with colliders. | Player object travels towards rigid object with collider and collides continually. | Player object continually is pushed away from the object it collides with, the player object remains upright and on a surface; player object does not glitch through other rigid objects with colliders. | Player object continually is pushed away from the object it collides with, the player object remains upright and on a surface; player object does not glitch through other rigid objects with colliders. | 🟩 |
| 8 | Disclaimer window appears with the according disclaimer text and disappears on on-click events. | Start button is pressed at the initial scene when the game is loaded. The disclaimer window button is then pressed if the disclaimer window appears. | The start button is pressed at the initial scene when the game is loaded, the disclaimer GUI window appears in the centre of the screen with the according text, when the disclaimer window (button) is pressed, the disclaimer window disappears and the proceeding scene is loaded. | The start button is pressed at the initial scene when the game is loaded, the disclaimer GUI window appears in the centre of the screen with the according text, when the disclaimer window (button) is pressed, the disclaimer window disappears and the proceeding scene is loaded. | 🟩 |

# Unit testing

## Purpose of unit testing

Opposing Blackbox testing, unit testing is a software testing technique which focuses on specific units or components within a given application; the code base of the given application is known, which is necessary to determine whether each unit within the applications functionality works as intended, in isolation. This is measured from a statistical basis, to ensure the accuracy of the tested units.

In relation to the mobile game as a piece of software, unit testing enables the identification of unit correctness or accuracy within application functionality. Therefore, unit testing is useful for determining numerical output correctness, specifically for the use of measuring forces applied to and acting on objects in the game, as well as pinpointing arithmetic and conditioning errors, which make up functionality within the application. Unit testing has been nominated as a testing method for its degree of precision that it provides for this matter.

### Unit testing cases

| Case | Summary | Process | Actual result(s) | Expected result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Determine whether the player object can only jump once, from when the corresponding button is pressed, and object is grounded | Interact with the jump button, whilst the player object is in the animation of jumping | 'isGrounded' returns 'false' when the player object is mid-air and returns 'true', when the player object is in contact with the grounding plane | 'isGrounded' returns 'false' when the player object is mid-air and returns 'true', when the player object is in contact with the grounding plane | 🟩 |
| 2 | Determine whether the subtitle sequence is only replayable after the current sequence coroutine finishes | Interact with the subtitle sequence button, whilst the subtitle sequence is active | 'morseCodeAudio' returns 'false' when the coroutine is active and returns 'true', when the coroutine has finished | 'morseCodeAudio' returns 'false' when the coroutine is active and returns 'true', when the coroutine has finished | 🟩 |
| 3 | Determine whether objects are only interactable with, when the first-person camera is enabled | Attempt to interact with interactable objects, while the third-person camera perspective is active | 'mousePressed' returns 'false' when the third-person camera is active and returns 'true', when the first-person camera is active | 'mousePressed' returns 'false' when the third-person camera is active and returns 'true', when the first-person camera is active | 🟩 |
| 4 | Determine whether the disclaimer GUI window closes upon clicking on it, as acknowledgment | Interact with the start and disclaimer GUI window buttons, in the menu screen scene | 'displayDisclaimer' returns 'true' when the disclaimer GUI window is initially made and returns 'false', when the disclaimer GUI window button is pressed upon | 'displayDisclaimer' returns 'true' when the disclaimer GUI window is made and returns 'false', when the disclaimer GUI window button is pressed upon | 🟩 |
| 5 | Determine whether one camera remains active and the other remains inactive, when the camera is toggled | Interact with the camera toggle button, embedded within the UI | 'firstCamera' returns 'true' when application starts and 'false' when cameras are toggled again. 'secondCamera' returns 'false' when application starts, and 'true' when cameras are toggled again. 'toggleCamera' returns 'true' and 'false' per alternating button press | 'firstCamera' returns 'true' when application starts and 'false' when cameras are toggled again. 'secondCamera' returns 'false' when application starts, and 'true' when cameras are toggled again. 'toggleCamera' returns 'true' and 'false' per alternating button press | 🟩 |

## Performance profiling

### Purpose of performance profiling

Performance profiling is a software testing technique that is particular to system performance, when a given system is executing software. Performance profiling has no intervention with an applications code base and like Blackbox testing, performance profiling can be observed visually, also. For which, performance profiling can be used to determine how systems perform in relation to responsiveness and stability, when executing software; a systems resource consumption and reliability can be measured.

In relation to the mobile game as a piece of software, performance profiling will allow the reliability of the software to be measured in systems, in the unit of frames per-second (FPS). This dictates the smoothness of the games representation and is dependent on the hardware of the executing system, the games visual representation, the number of events and objects that appear in a scene and the interactions that occur between these objects. In which, higher displaying units of FPS indicate a better system performance. I have nominated performance profiling, as a testing method to determine the playability of the game, over the duration of its development.

### Performance profile test cases

| Case | Summary | Process | Actual result(s) |
|---|---|---|---|
| 1 | First scene, start-up performance profiling | Load into the first scene of the game, capture performance using Unity profiler | Peak usage: CPU (97.5%), GPU (91.4%), RAM (1.55GB)<br>Minimum usage: CPU (96.2%), GPU (13.4%), RAM (1.49GB)<br>Average FPS: 145.58 |
| 2 | First scene, player exploring environment performance profiling | Load into the first scene of the game and explore environment, capture performance using Unity profiler | Peak usage: CPU (98.4%), GPU (95.9%), RAM (1.82GB)<br>Minimum usage: CPU (96.3%), GPU (17.5%), RAM (1.71GB)<br>Average FPS: 141.70 |
| 3 | First scene, switching camera perspectives performance profiling | Load into the first scene of the game and switch camera perspectives continuously, capture performance using Unity profiler | Peak usage: CPU (97.3%), GPU (85.8%), RAM (1.91GB)<br>Minimum usage: CPU (97.2%), GPU (21.6%), RAM (1.79GB)<br>Average FPS: 137.23 |
| 4 | Loading screen scene, start-up performance profiling | Load game from menu screen, capture performance using Unity profiler | Peak usage: CPU (97.8%), GPU (29.4%), RAM (1.18GB)<br>Minimum usage: CPU (97.0%), GPU (10.8%), RAM (1.17GB)<br>Average FPS: 145.61 |

***Appendix D:***

# System overview

'Under Lock and Key' is a mobile-based game that aims to provide players with a three-dimensional outlook on solving puzzles, as a game bound by the puzzle genre. Players of the game are expected to be able to view the game from within first-person and third-person perspectives, to help solve the games puzzles; the basis of the game will require players to be able to interact with objects throughout various house-type environments, in the form of rooms. A puzzle should be represented as a room, in which, the objects that lie within each room should be components of the puzzle. To complete all of the game's puzzles, players are required to transition between each of the rooms. For which, when each of the room's objectives can be identified as complete, the player should be able to proceed and therefore complete the game.

# System Design Document (SDD)

## SDD overview

This SDD purposes to describe the reporting systems requirements, system architecture, format of input and user interface (UI) design, for which the format of input will address. All of which information, is necessary for navigating the development and implementation of the system, programmatically. In the context of the project, the intended audience of this document aims to be for the project manager and developer(s). Moreover, in relation to the systems requirements, there should be correspondence with the systems functional requirements.

## System design

'Under Lock and Key' aims to enable players to be involved within an immersive gaming experience that adopts conventions of mystery and suspense to enhance the narrative plot of the game; for which, is based upon a prisoner with only one objective, to escape. The systems design purposes to enable the narrative of the game, where a player can interact with and resolve puzzles progressively. A single system is sufficient for this purpose.

**Design objectives**

- Camera object control
- Player object control
- Player interaction
- Puzzle sequencing
- Scene transitioning
- Navigable configuration menus
- Subtitling
- Disclaimer prompt
- UI addressing control and interaction
- Conventional to puzzle genre
- Stable system performance

In relation to the systems intended audience, it is necessary for users of the system to be advised about the nature of the games content, as themes of suspense are intended to be illustrated aesthetically and portrayed as sinister. The systems design makes considerations for player discretion, through implementing disclaimer-related prompts; this aims to inform all age of player, but especially youthful players, who are considered more prone to being scared.

## Design assumptions

The systems availability will exist across android mobile platforms only, for which, the system will be developed and deployed for android version 4.1 (Lollipop) and above devices; this supports the majority of android user devices. Throughout the development of the game, changes and alterations to the systems functionality and arrangement are probable to occur, mostly as the result of testing and debugging. In which, development and testing processes of the system are likely to be conducted simultaneously, attempting to achieve proper functionality and robustness of the systems code base, progressively. System development will attempt to adhere to the Agile methodology SCRUM; testing, exploratory. Moreover, users of the system will be players of the game, who will be able to complete the games objectives through interacting with a mobile devices interface. Users will not require an active internet connection to use the system, therefore there will be no network considerations.

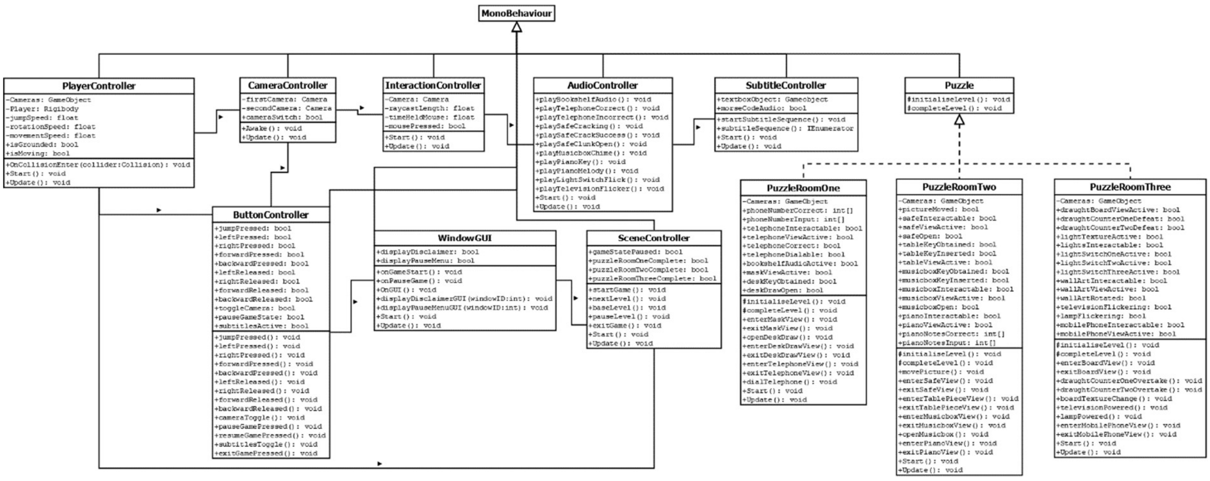
## Design constraints

The systems design will present increased focus upon the implementation of game functionality, rather than performance considerations; however, significant performance issues will be accounted for throughout the games development. Thereby, there will be a trade-off between the quantity of functionality and quality of performance. In further correspondence to testing, there is a lack of devices available to analyse the systems performance effectively; an array of devices is required and so the systems performance cannot be assured across numerous devices. Due to the potential longevity of the game, the design will only consider multiple levels of the game, as the result of time restraints and lack of development and testing personnel. Also, in scope of time restraints, as the systems development is time dependant, the games aesthetic and some narrative intentions may not be delivered; this could also be anticipated by a lack of third-party resources available, to aid the games development.

## System architecture

Relating to the systems architecture, I have opted for an object-orientated programming (OOP) approach for the use of classes, for which, each of said classes proposes to separate and group the systems attributes and behaviours in the form of data and functions, by association. From the application of classes, the system aims to be organised structurally and as a commonly used data structure within previous developments, the application of classes assumes a hastened rate of development. Additionally, the system proposes to adopt the principle of inheritance between classes, for the means of interacting with and overriding base class members; this exemplifies the concept of polymorphism and is useful for classes that have similar functionality. Furthermore, the system will also make use of encapsulation from the existence of class member access modifiers, these govern the accessibility of class members to external classes; access modifiers will allow classes of the system to provide protection from accidental member modification, from other classes residing in the system.

In the demonstration of the proposed systems architecture, I have constructed a Unified Model Language (UML) diagram for the purpose of representing the systems structural design. In which, the diagram aims to acknowledge the relationship between the systems classes and to identify each of the classes data and functional members. From the existence of the UML class diagram, the development of the system can be navigated to achieve the functional requirements of the game and can therefore fasten the development process. However, as previously mentioned the arrangement of the systems structure and its accompanying functionality are subject to change; therefore, the diagram presented below should be interpreted as the preliminary system design.



## User Interface

Designing the in-game user interface (UI) requires the support for players being able to interact with objects, within their environments throughout the game; the interface should not be a restraint on a player's ability to complete game objectives. In accordance with the functional requirements of the game, the player-controlled object should be able to do the following:

- The player-controlled game object can traverse in the facing direction of the active camera object, when the corresponding user interface buttons are interacted with
- The player-controlled game objects facing direction can be adjusted, when the corresponding user interface buttons are interacted with

- The player-controlled game object can interact with interactable objects, when within the required proximity of these objects, through touch, hold and swipe interactions

In relation to player-controlled object movement, the player should be able to traverse forwards and backwards in the facing direction of the camera. Also, the player-controlled object should be able to rotate around its own axis, universally, to adjust its facing and traversing direction. This functionality can be addressed by the use of buttons, when pressed or held; the buttons should be cast aside from the central viewpoint of the active camera's projection, this aims to prevent view obstruction. Moreover, relating to object interaction, players can be abled to engage with interactable objects via performing touch, hold or swipe gestures upon the playing devices interface. This can be achieved through the implementation of ray casting, which does not require the existence of buttons.

Furthermore, to aid players with objective and puzzle resolution, alternating camera perspectives should be implemented to enhance the players field of view (FOV) and environmental awareness. Relating back to the game's functional requirements, the games scenes should enable the following:

- The scenes will allow the presence of numerous player-controlled camera objects

Similarly, alternating camera perspectives can be addressed via the existence of a button, when pressed. The button controlling the functionality should also be positioned on the edging perimeter of a user's device, to prevent view obstruction. However, to prevent accidental interaction, the button should also be positioned uniquely from player-controlled movement buttons. In the following illustration the discussed functionality can be conceptually visualised and addressed, in consideration of the intended orientation of user device, landscape.

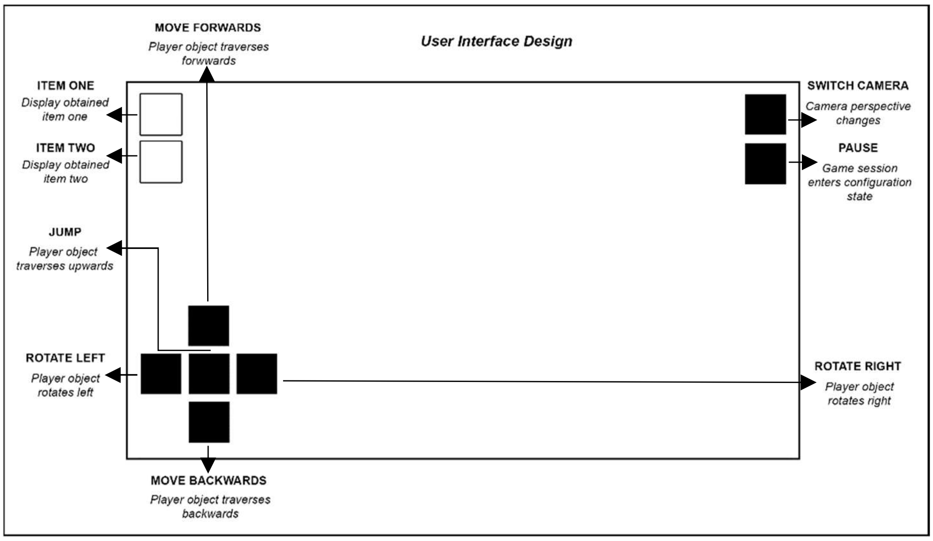

MOVE FORWARDS
Player object traverses forwwards
User Interface Design
ITEM ONE
Display obtained item one
ITEM TWO
Display obtained item two
SWITCH CAMERA
Camera perspective changes
PAUSE
Game session enters configuration state
JUMP
Player object traverses upwards
ROTATE LEFT
Player object rotates left
ROTATE RIGHT
Player object rotates right
MOVE BACKWARDS
Player object traverses backwards

*Figure 1: In-game UI design, when the game session is in a playable state*

Moreover, a player should be made able to pause the game sessions state and adjust the games configuration when within said state, as detailed within the game's functional requirements:

- The application will allow a player to pause the state of a game session

Pausing the state of a game session can also be addressed via the application of a button, when pressed. Furthermore, the button can also be positioned near to the alternating camera perspective button, which attempts to enable the interface to be compact and easily navigable. Unlike player-controlled movement, alternating camera perspectives and entering player configuration menus are assumed to be interacted with less, which insinuates that this design choice is suitable as players will be encouraged to press these buttons consciously; thus, avoiding accidental interaction. As expected from entering the paused state of the game session, the system should facilitate the following:

- The application will allow a player to toggle the active state of subtitling
- The application will allow a player to exit a game session

In the following image, the intended functionality of the pause menu interface can be conceptually visualised and addressed.

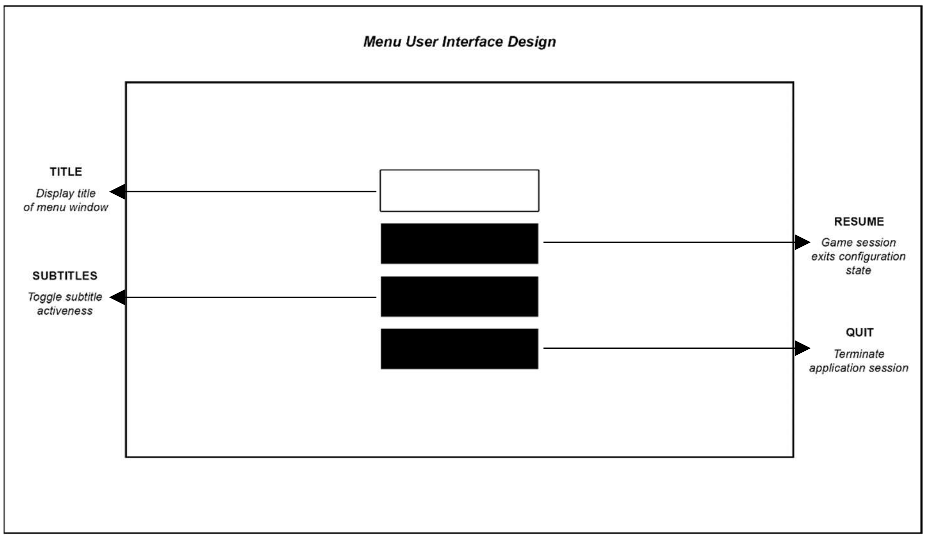

Menu User Interface Design
TITLE
Display title of menu window
SUBTITLES
Toggle subtitle activeness
RESUME
Game session exits configuration state
QUIT
Terminate application session

*Figure 2: Menu UI design, when the game session enters the configuration state*

In addition to fulfilling the functional requirements of the proposed system, the in-game UI design also aims to enable players to identify items they have obtained, from their interactions in each puzzle room scene. In accordance to the game's objectives, items in the form of game objects are proposed to be obtained by a player and used in conjunction with other objects when interacted with; from a player being able to acknowledge that it has possession of items, aims to navigate a player to its next objective in the sequence. In which, the way an item appears visually attempts to familiarise players with a related game object in the surrounding environment, therefore, the next objective or puzzle component becomes more obvious to the player. Summarily, said technique will enable gradual gameplay progression and will further try to prevent player frustration from arising.

Player acknowledgement regarding item possession can be addressed via the application of images corresponding to the game objects a player obtains. Such images should be placed and positioned away from all of buttons previously specified but should reside within the edging perimeter of a user's device. This design choice aims to prevent accidental interaction with buttons and to maintain the compactness and comprehensive state of the interface, also.

Meanwhile, in relation to the interface of the loading screen scene, a player should be able to start a game session as specified within the functional requirements:

- The application will allow a player to start a game session

As seen within all of the previously illustrated interfaces, the ability to start a game session will also be addressed via the implementation of a button, when pressed. However, unlike the in-game UI design, there is no requirement for specific button positioning as an active game session would not exist. In which, the arrangement chosen for the loading screen UI imposes to be unique in comparison to the other UI designs, whereby it is considered aesthetically driven. In correlation to the pause menu UI design, the loading screen UI design also addresses the following functionality:

- The application will allow a player to toggle the active state of subtitling
- The application will allow a player to exit a game session

In consideration of the interface's functionality, the diagram below attempts to exemplify the visual and functional aspects of the interface.

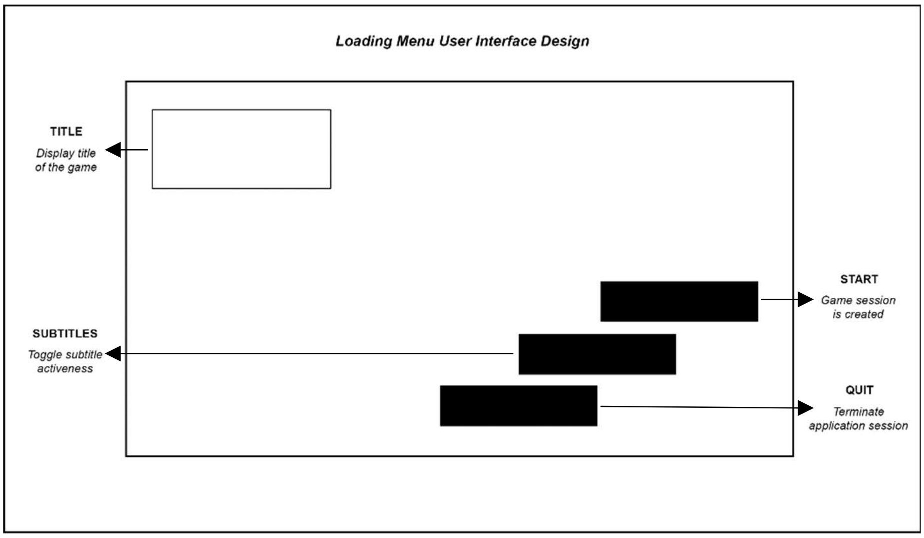

Loading Menu User Interface Design
TITLE
Display title of the game
START
Game session is created
SUBTITLES
Toggle subtitle activeness
QUIT
Terminate application session

*Figure 3: Loading screen UI design, when the application initially loads into memory*

***Appendix E:***

**Introduction (max. 100 words):**

This document purposes to provide insight into the development of my project, a 3D puzzle game titled 'Under Lock and Key', which situates as a 'escape the room' type game; to note, this project will utilise Unity game engine for its development. The basis of the game follows the narrative of the player who is captured and taken to a house with interconnected rooms containing puzzles. The players objective of the game is to interact with their environment with use of basic numeracy and physical skills, to solve each puzzle; for each puzzle that is complete, the player nears escaping.

**Project Background (max. 300 words): A brief description providing the project background/context. e.g. is it based on a business need? a technical need? does it arise from the interests of a particular person/company?**

'Under Lock and Key' is a common saying within British society and infers to an individual or group of people who are being kept in a place, generally a place which is inescapable; usually being a prison. As a 3D puzzler game, I have chosen to adopt an 'escape the room' orientation of gameplay, as inspired by the movie series 'Saw' and mobile game series 'The Room'. For which the name 'Under Lock and Key', enforces the theme of imprisonment I am aiming to portray for the game; and as imagined in the film and game series mentioned, the concept of imprisonment is created from the use of rooms containing puzzles. Inevitably this is an idea I want to be prominent throughout the game and am proposing it to be addressed from the existence and interaction of models, lighting, cameras, textures, audio and buttons within a user interface which are bound together with scripts.

Notably the game bases itself upon the interior of a large house/ building and as the narrative begins, the player is captured and considered a prisoner in the house, unless he/ she can complete all the puzzles in the corresponding rooms and escape.

Moreover, in further relation to the said film and game series, the game aesthetics I want to apply are also influenced by the sinister-like environments seen. Whereby more like 'The Room' game series, low-key lit environments are seamlessly used to present darker tones and colours; from observation, low-key lit environments dramatically increase the level of suspense of a player's experience. Suspense is another theme I want to adopt into the game given its puzzler-nature, which aims to slow the pace of gameplay and for players to attempt to complete puzzles methodically, rather than to approach them illogically and aimlessly.

**Aims (max. 100 words): A statement of the overall aims of the project.**

The project aims to present a prototype version of a 3D puzzler game, developed for mobile specific use using Unity game engine. Within the game, the game aims to present a player with puzzles within their relative environment. Each puzzle is intended to employ a players numerical and or physical capabilities at a basic level; and are intended to be addressed by the existence of models, lighting, cameras, textures, audio and buttons within a user interface which are bound together with scripts. A level should be made to display the game mechanics and puzzle sequencing, featuring numerous interconnected rooms.

**Objectives (max. 200 words): A list of specific, measurable objectives, each of which is likely to result in a deliverable. They specify all the work tasks to be undertaken to meet the stated aim. They will vary from project to project, as every project is different, but some examples are provided below. All projects will need to review and report on the literature in a chosen area. Projects might include such general objectives as: To investigate system requirements and produce a Requirements Specification. To research and write a report on good practice in HCI design. To design an interface using the findings from the HCI report. To design and execute a suitable test plan. Or they might be more specific, e.g.: To review and report on how mathematical simulation techniques could be applied to a traffic simulator.**

- To create a literature review document with a word count of 2000 words regarding the content of existing 3D puzzler games
- To create a functional requirements document detailing the operations and activities (functionality) of the 3D puzzler games system
- To create a suitable indicative test plan document, which discusses the methods of testing and the test cases conducted (actual test cases)
- To create a system design document exploring the 3D puzzler game system architecture, user interface design and creating UML/ ERD diagrams to show the relation between classes
- To create an implementation report document with a word count of 300 words regarding what the prototype demonstration of the 3D puzzler game will include (front-end (visual) and back-end (code))
- To create a main report document with a word count of 8000 words presenting the system functionality implementation and decisions, development of software lifecycle methodology and system analysis and reflection
- To attend a viva meeting, demonstrating created system and handling questions that follow professionally
- To create a 3D puzzler game in accordance with my agreed upon project contract objectives
- To attend a minimum of 10 management meetings with my allocated supervisor

**Deliverables (max. 100 words): A list of your Project's deliverables with some general description could be found in the module specification.**

First deliverable (25%) - due 10/01/2020 (3:00 pm)

- Literature review (20%)
- Functional requirements (20%)
- Indicative test plan (20%)
- System design documentation (20%)
- Implementation report (20%)

Final deliverable (75%) - due 1/5/2020 (3:00 pm)

- Main report (35%)
- Viva (15%)
- The system (40%)
- Management meetings (10%)

**Resources and Constraints (max. 100 words): A list of any specific resources that the project requires; for example, hardware and software; access to people or organisations. A list of any known constraints, for example, availability of certain resources.**

Hardware

- Computer device (laptop, personal desktop, or University desktop device), access and availability of computer labs on University campus is required if use is needed
- USB device (portability of project)

Software

- Unity game engine (game development)
- Microsoft Visual Studio (game scripting)
- Adobe Photoshop (game textures/ materials, GUI)
- Audacity (game audio effects)
- Microsoft Office - Word (written deliverables and general note taking)
- Internet client (research and project implementation guidance)

Constraints

- Internet availability for submission of deliverables
- Personal computing devices must be used in the case of lacking availability of computer labs on University campus with the suite of software I require

**Sources of Information (max. 100 words): A list of sources you intend to use. These could include: Specific books/journals if you already know of them; Library/Internet; Organisations or individuals you intend to contact.**

Internet

- The internet will serve as my predominant method for sourcing information, the internet can provide information and documentation regarding program code implementation, existing games like that of 'The Room' series, films like that of the 'Saw' series and other media to inform me about existing mechanics, aesthetics and gameplay; inspiring me with ideas for my own project. Game assets can also be acquired from relevant sites to aid the games development.

Unity asset store

- In addition to the internet, I can gather assets from the asset store within the Unity game engine, as a point of convenience.

**Risk Analysis (max. 100 words): What could endanger your project, what will you do if it happens.**

Throughout the creation of my project, a series of computing devices will be used for its creation; University owned and personally owned. Computer devices are known to malfunction, corrupt stored data or be prone to physical damage at random or in the event of system failures. To overcome this, local copies of the project will be made on a regular basis; University devices have profiles which are cloud-based and are not of concern. To prevent the loss of my project, I am going to upload my project regularly to a cloud-based service, OneDrive, ensuring that my project will always exist.

**Schedule of Activities (max. 300 words): Having defined the tasks to be undertaken in the list of objectives, you need to prepare a Project Plan to show how you intend to carry them out.**

For conducting my tasks which make up the objectives for the project, I have created a Gantt chart which aims to visually indicate the order in which the objectives will be complete and their indicative time necessary to be completed. Completion is measured in days, but accounts for the typical workable hours within a day; being eight hours. This measure considers the timing of sleeping (eight hours), food consumption related activities (two hours), hygiene maintenance related activities (one hour), the most congested University timetabled day (five hours) and minor intricacies that may occur between the transitioning from and to any said time-based event. To make note, weekends are excluded as workable days for the project, this is due to work commitments, for which I am legally required to fulfil sixteen hours of work over this period, every week.

As set out and in accordance with the Gantt chart, objectives are conducted chronologically in correspondence to their deadlines and issued order; the order being: student forms, first deliverables and lastly the final deliverables. The overall time required to complete the project from the starting work effort (14/10/19) to the ending work effort (18/02/20) is fifty-seven days (indicative), which suffices for approximately 1.9 months or 456 working hours. For the viva, the chart does not consider the actual date of it happening as it is not given and assumes it proceeds from all deliverables being submitted; the chart shows the time necessary to prepare for and deliver the viva. Similarly, the actual dates of each project progress meeting are not considered as they are not pre-determined, but the time accumulated for all meetings is considered and shown within the chart. It is without doubt that the game will require the most time for its completion given its scale, and project grade weighting.

***Appendix F:***

**Please indicate which of these possible attributes is addressed by your undertaking of this project. You should select at least two items.**

| | Addressed by Project? |
|---|---|
| 1- Ability to work collaboratively: teams from a range of backgrounds and countries | No |
| 2- Excellent communication skills with a sensitivity to speaking with and listening to non-native English speakers | No |
| 3- An ability to embrace multiple perspectives and challenge thinking in a range of cultural context | No |
| 4- A capacity to develop new skills and behaviours according to role requirements | **Yes** |
| 5- An ability to negotiate and influence clients across the globe from different cultures | No |

| | |
|---|---|
| 6- An ability to form professional, global networks | No |
| 7- An openness to/respect of a range of perspectives from around the world | No |
| 8- Multi-cultural learning agility (i.e. able to learn in any culture or environment) | **Yes** |

**Brief description of how the ticked attributes have been addressed (max. 50 words per item):**

**Item 4:** As each puzzle will have multiple objectives, which totals to its completion, players will think and approach to complete puzzles non-sequentially, encouraging players to recognise an order and figure ways to interact with their environment to complete said puzzles; puzzles employ basic numeracy and physical skills for their completion.

**Item 8:** The game will feature minimal amounts of text displaying to the screen; audio will be specific to an objective which makes up a puzzle. Audio will be subtitled in numeric (or Morse) format to enable the completion of puzzles, no matter a player's known languages. Players require a mobile device.

*Appendix G:*

**Brief description of proposed activity and its objectives**

- A 3D puzzler game specific to mobile devices, the game establishes a sinister themed environment excluding gore and violent elements to create a suspenseful gameplay experience. The game 'Under Lock and Key' proposes as a 'escape the room' game type, which follows the narrative of a girl who is captured and taken to a house with interconnected rooms containing puzzles. The players objective of the game is to interact with their environment with use of basic numeracy and physical skills, to solve the given puzzles. With each puzzle complete, nears the players escape from the imprisonment.

**Project objectives**

- Create a 3D puzzler game which is played on mobile devices
- Create sinister themed world space, which is maintained throughout the game, with consideration of evading age/ audience restriction (use of textures, lighting and materials to create desired aesthetic)
- Creation and use of camera(s) to enhance player awareness and experience of solution orientation of gameplay
- Incorporate the use of models, lighting, textures (materials), cameras, audio and buttons within a user interface, which should be bound together with scripts to address the puzzles and their progress of completion
- Create a level to display the game mechanics and puzzle sequencing (narrative development)

**Ethical Issues Identified**

- As the proposed aesthetic of the game is going to adopt a sinister themed environment for the player to explore, players of younger age groups may interpret such environment to be scary, even when any thriller elements are not apparent. The sinister theme is to be achieved from using dull colouring and low intensity/ lack of lighting to simulate dimness; this aims to create my desired suspense of gameplay.
- The game will require someone to test for working functionality of game mechanics and the sequencing of gameplay

**How these will be addressed**

- Overcoming age restricted content, the game will not feature content which aims to scare and or portray gore and violence; this will satisfy younger audiences. Moreover, the loading title/ menu screen of the game will present the games aesthetic, which aims to introduce the suspense of gameplay and sinister themed environment the player can expect to be in. The proposed rating for the game will be ‘Everyone’ in accordance with ESRB rating guides, but a disclaimer will be made visible at the menu screen upon loading the game.
- Overcoming testing, I myself will be testing the game throughout its development cycle, which enables improper functionality to be amended when discovered

***Appendix H:***

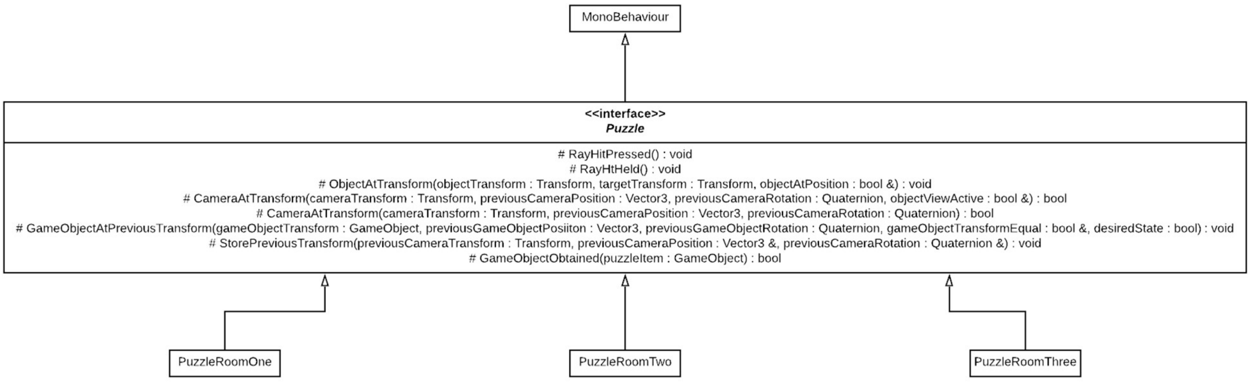

MonoBehaviour
<<interface>>
Puzzle
# RayHitPressed() : void
# RayHtHeld() : void
# ObjectAtTransform(objectTransform : Transform, targetTransform : Transform, objectAtPosition : bool &) : void
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion, objectViewActive : bool &) : bool
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion) : bool
# GameObjectAtPreviousTransform(gameObjectTransform : GameObject, previousGameObjectPosiiton : Vector3, previousGameObjectRotation : Quaternion, gameObjectTransformEqual : bool &, desiredState : bool) : void
# StorePreviousTransform(previousCameraTransform : Transform, previousCameraPosition : Vector3 &, previousCameraRotation : Quaternion &) : void
# GameObjectObtained(puzzleItem : GameObject) : bool
PuzzleRoomOne
PuzzleRoomTwo
PuzzleRoomThree

*Figure 62: ‘Puzzle’ abstract class, contain the virtual methods to be implemented within the puzzle room classes, display the inheritance relation between the puzzle room classes and their interface*

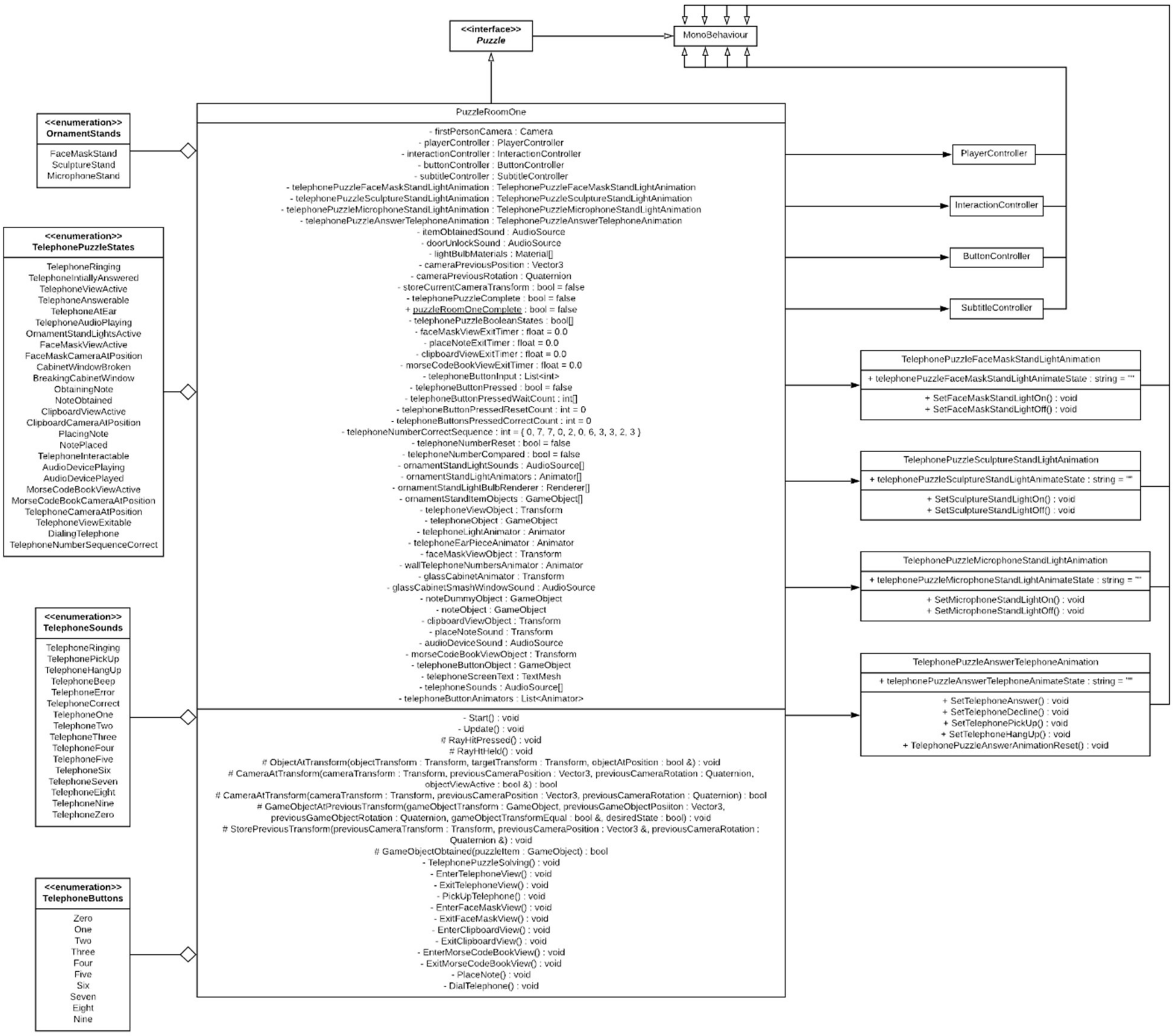
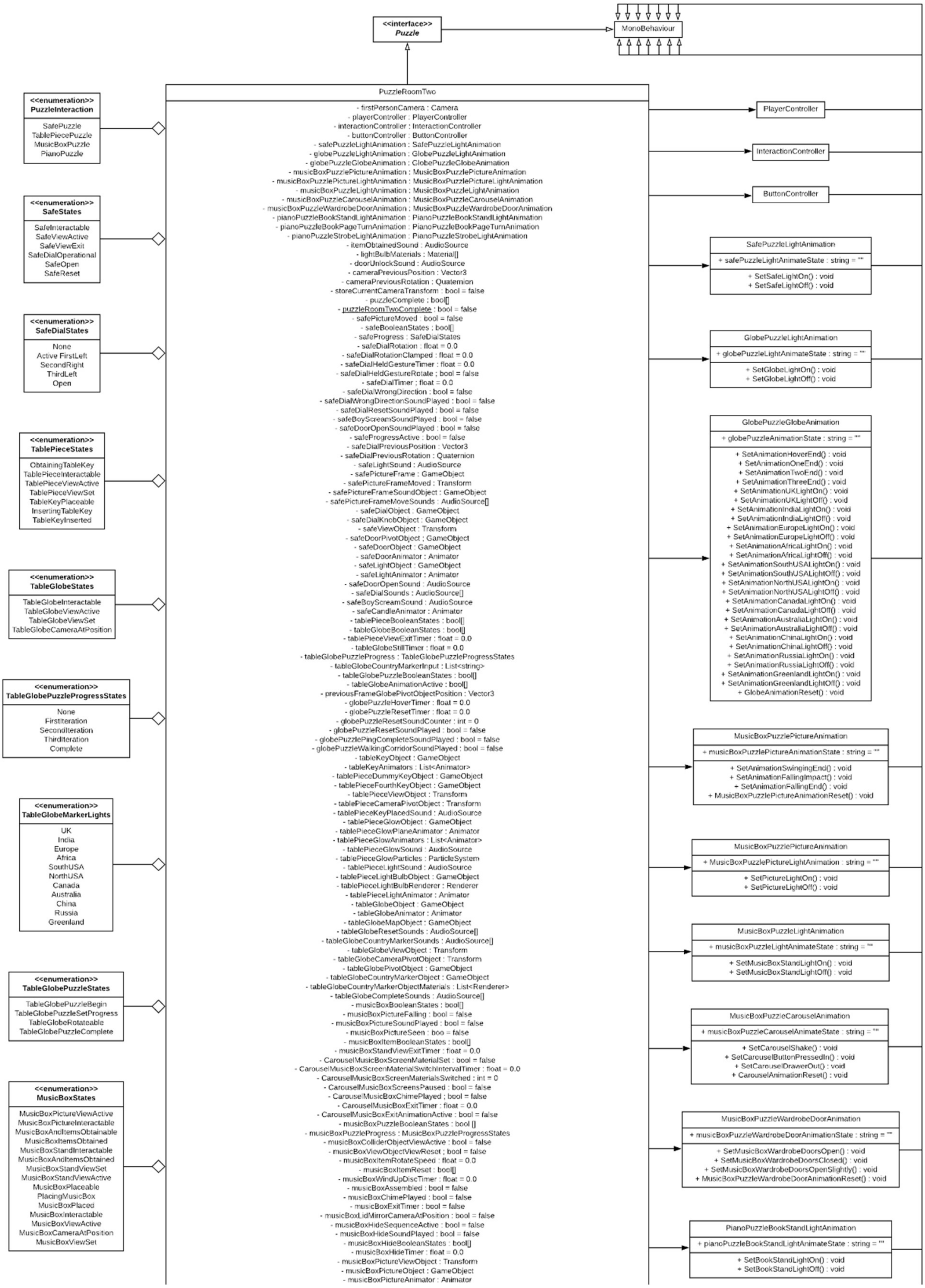
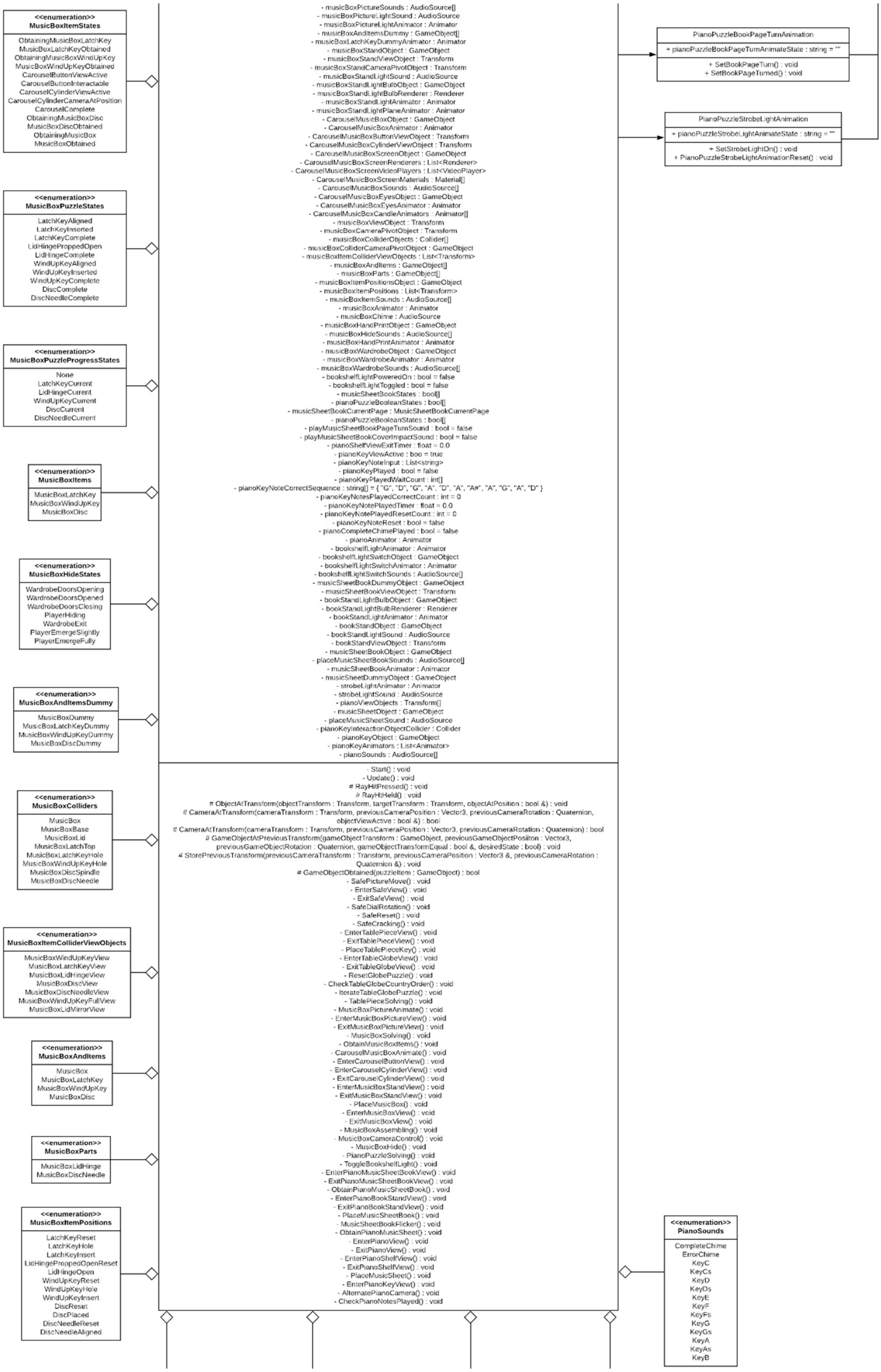

<<interface>>
Puzzle
MonoBehaviour
PuzzleRoomOne
- firstPersonCamera : Camera
- playerController : PlayerController
- interactionController : InteractionController
- buttonController : ButtonController
- subtitleController : SubtitleController
- telephonePuzzleFaceMaskStandLightAnimation : TelephonePuzzleFaceMaskStandLightAnimation
- telephonePuzzleSculptureStandLightAnimation : TelephonePuzzleSculptureStandLightAnimation
- telephonePuzzleMicrophoneStandLightAnimation : TelephonePuzzleMicrophoneStandLightAnimation
- telephonePuzzleAnswerTelephoneAnimation : TelephonePuzzleAnswerTelephoneAnimation
- itemObtainedSound : AudioSource
- doorUnlockSound : AudioSource
- lightBulbMaterials : Material[]
- cameraPreviousPosition : Vector3
- cameraPreviousRotation : Quaternion
- storeCurrentCameraTransform : bool = false
- telephonePuzzleComplete : bool = false
+ puzzleRoomOneComplete : bool = false
- telephonePuzzleBooleanStates : bool[]
- faceMaskViewExitTimer : float = 0.0
- placeNoteExitTimer : float = 0.0
- clipboardViewExitTimer : float = 0.0
- morseCodeBookViewExitTimer : float = 0.0
- telephoneButtonInput : List<int>
- telephoneButtonPressed : bool = false
- telephoneButtonPressedWaitCount : int[]
- telephoneButtonPressedResetCount : int = 0
- telephoneButtonsPressedCorrectCount : int = 0
- telephoneNumberCorrectSequence : int = { 0, 7, 7, 0, 2, 0, 6, 3, 3, 2, 3 }
- telephoneNumberReset : bool = false
- telephoneNumberCompared : bool = false
- ornamentStandLightSounds : AudioSource[]
- ornamentStandLightAnimators : Animator[]
- ornamentStandLightBulbRenderer : Renderer[]
- ornamentStandItemObjects : GameObject[]
- telephoneViewObject : Transform
- telephoneObject : GameObject
- telephoneLightAnimator : Animator
- telephoneEarPieceAnimator : Animator
- faceMaskViewObject : Transform
- wallTelephoneNumbersAnimator : Animator
- glassCabinetAnimator : Transform
- glassCabinetSmashWindowSound : AudioSource
- noteDummyObject : GameObject
- noteObject : GameObject
- clipboardViewObject : Transform
- placeNoteSound : Transform
- audioDeviceSound : AudioSource
- morseCodeBookViewObject : Transform
- telephoneButtonObject : GameObject
- telephoneScreenText : TextMesh
- telephoneSounds : AudioSource[]
- telephoneButtonAnimators : List<Animator>
- Start() : void
- Update() : void
# RayHitPressed() : void
# RayHtHeld() : void
# ObjectAtTransform(objectTransform : Transform, targetTransform : Transform, objectAtPosition : bool &) : void
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion, objectViewActive : bool &) : bool
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion) : bool
# GameObjectAtPreviousTransform(gameObjectTransform : GameObject, previousGameObjectPosiiton : Vector3, previousGameObjectRotation : Quaternion, gameObjectTransformEqual : bool &, desiredState : bool) : void
# StorePreviousTransform(previousCameraTransform : Transform, previousCameraPosition : Vector3 &, previousCameraRotation : Quaternion &) : void
# GameObjectObtained(puzzleItem : GameObject) : bool
- TelephonePuzzleSolving() : void
- EnterTelephoneView() : void
- ExitTelephoneView() : void
- PickUpTelephone() : void
- EnterFaceMaskView() : void
- ExitFaceMaskView() : void
- EnterClipboardView() : void
- ExitClipboardView() : void
- EnterMorseCodeBookView() : void
- ExitMorseCodeBookView() : void
- PlaceNote() : void
- DialTelephone() : void
<<enumeration>>
OrnamentStands
FaceMaskStand
SculptureStand
MicrophoneStand
<<enumeration>>
TelephonePuzzleStates
TelephoneRinging
TelephoneIntiallyAnswered
TelephoneViewActive
TelephoneAnswerable
TelephoneAtEar
TelephoneAudioPlaying
OrnamentStandLightsActive
FaceMaskViewActive
FaceMaskCameraAtPosition
CabinetWindowBroken
BreakingCabinetWindow
ObtainingNote
NoteObtained
ClipboardViewActive
ClipboardCameraAtPosition
PlacingNote
NotePlaced
TelephoneInteractable
AudioDevicePlaying
AudioDevicePlayed
MorseCodeBookViewActive
MorseCodeBookCameraAtPosition
TelephoneCameraAtPosition
TelephoneViewExitable
DialingTelephone
TelephoneNumberSequenceCorrect
<<enumeration>>
TelephoneSounds
TelephoneRinging
TelephonePickUp
TelephoneHangUp
TelephoneBeep
TelephoneError
TelephoneCorrect
TelephoneOne
TelephoneTwo
TelephoneThree
TelephoneFour
TelephoneFive
TelephoneSix
TelephoneSeven
TelephoneEight
TelephoneNine
TelephoneZero
<<enumeration>>
TelephoneButtons
Zero
One
Two
Three
Four
Five
Six
Seven
Eight
Nine
PlayerController
InteractionController
ButtonController
SubtitleController
TelephonePuzzleFaceMaskStandLightAnimation
+ telephonePuzzleFaceMaskStandLightAnimateState : string = ""
+ SetFaceMaskStandLightOn() : void
+ SetFaceMaskStandLightOff() : void
TelephonePuzzleSculptureStandLightAnimation
+ telephonePuzzleSculptureStandLightAnimateState : string = ""
+ SetSculptureStandLightOn() : void
+ SetSculptureStandLightOff() : void
TelephonePuzzleMicrophoneStandLightAnimation
+ telephonePuzzleMicrophoneStandLightAnimateState : string = ""
+ SetMicrophoneStandLightOn() : void
+ SetMicrophoneStandLightOff() : void
TelephonePuzzleAnswerTelephoneAnimation
+ telephonePuzzleAnswerTelephoneAnimateState : string = ""
+ SetTelephoneAnswer() : void
+ SetTelephoneDecline() : void
+ SetTelephonePickUp() : void
+ SetTelephoneHangUp() : void
+ TelephonePuzzleAnswerAnimationReset() : void

*Figure 63: 'PuzzleRoomOne' class, illustrating the several instances of association to external classes and enumerations*



<<interface>>
Puzzle
MonoBehaviour
PuzzleRoomTwo
- firstPersonCamera : Camera
- playerController : PlayerController
- interactionController : InteractionController
- buttonController : ButtonController
- safePuzzleLightAnimation : SafePuzzleLightAnimation
- globePuzzleLightAnimation : GlobePuzzleLightAnimation
- globePuzzleGlobeAnimation : GlobePuzzleGlobeAnimation
- musicBoxPuzzlePictureAnimation : MusicBoxPuzzlePictureAnimation
- musicBoxPuzzlePictureLightAnimation : MusicBoxPuzzlePictureLightAnimation
- musicBoxPuzzleLightAnimation : MusicBoxPuzzleLightAnimation
- musicBoxPuzzleCarouselAnimation : MusicBoxPuzzleCarouselAnimation
- musicBoxPuzzleWardrobeDoorAnimation : MusicBoxPuzzleWardrobeDoorAnimation
- pianoPuzzleBookStandLightAnimation : PianoPuzzleBookStandLightAnimation
- pianoPuzzleBookPageTurnAnimation : PianoPuzzleBookPageTurnAnimation
- pianoPuzzleStrobeLightAnimation : PianoPuzzleStrobeLightAnimation
- itemObtainedSound : AudioSource
- lightBulbMaterials : Material[]
- doorUnlockSound : AudioSource
- cameraPreviousPosition : Vector3
- cameraPreviousRotation : Quaternion
- storeCurrentCameraTransform : bool = false
- puzzleComplete : bool[]
- puzzleRoomTwoComplete : bool = false
- safePictureMoved : bool = false
- safeBooleanStates : bool[]
- safeProgress : SafeDialStates
- safeDialRotation : float = 0.0
- safeDialRotationClamped : float = 0.0
- safeDialHeldGestureTimer : float = 0.0
- safeDialHeldGestureRotate ; bool = false
- safeDialTimer ; float = 0.0
- safeDialWrongDirection : bool = false
- safeDialWrongDirectionSoundPlayed : bool = false
- safeDialResetSoundPlayed : bool = false
- safeBoyScreamSoundPlayed : bool = false
- safeDoorOpenSoundPlayed : bool = false
- safeProgressActive : bool = false
- safeDialPreviousPosition : Vector3
- safeDialPreviousRotation : Quaternion
- safeLightSound : AudioSource
- safePictureFrame : GameObject
- safePictureFrameMoved : Transform
- safePictureFrameSoundObject : GameObject
- safePictureFrameMoveSounds : AudioSource[]
- safeDialObject : GameObject
- safeDialKnobObject : GameObject
- safeViewObject : Transform
- safeDoorPivotObject ; GameObject
- safeDoorObject : GameObject
- safeDoorAnimator : Animator
- safeLightObject : GameObject
- safeLightAnimator : Animator
- safeDoorOpenSound : AudioSource
- safeDialSounds : AudioSource[]
- safeBoyScreamSound : AudioSource
- safeCandleAnimator : Animator
- tablePieceBooleanStates : bool[]
- tableGlobeBooleanStates : bool[]
- tablePieceViewExitTimer : float = 0.0
- tableGlobeStillTimer : float = 0.0
- tableGlobePuzzleProgress : TableGlobePuzzleProgressStates
- tableGlobeCountryMarkerInput : List<string>
- tableGlobePuzzleBooleanStates : bool[]
- tableGlobeAnimationActive : bool[]
- previousFrameGlobePivotObjectPosition : Vector3
- globePuzzleHoverTimer : float = 0.0
- globePuzzleResetTimer : float = 0.0
- globePuzzleResetSoundCounter : int = 0
- globePuzzleResetSoundPlayed : bool = false
- globePuzzlePingCompleteSoundPlayed : bool = false
- globePuzzleWalkingCorridorSoundPlayed : bool = false
- tableKeyObject : GameObject
- tableKeyAnimators : List<Animator>
- tablePieceDummyKeyObject : GameObject
- tablePieceFourthKeyObject : GameObject
- tablePieceViewObject : Transform
- tablePieceCameraPivotObject : Transform
- tablePieceKeyPlacedSound : AudioSource
- tablePieceGlowObject : GameObject
- tablePieceGlowPlaneAnimator : Animator
- tablePieceGlowAnimators : List<Animator>
- tablePieceGlowSound : AudioSource
- tablePieceGlowParticles : ParticleSystem
- tablePieceLightSound : AudioSource
- tablePieceLightBulbObject : GameObject
- tablePieceLightBulbRenderer : Renderer
- tablePieceLightAnimator : Animator
- tableGlobeObject : GameObject
- tableGlobeAnimator : Animator
- tableGlobeMapObject : GameObject
- tableGlobeResetSounds : AudioSource[]
- tableGlobeCountryMarkerSounds : AudioSource[]
- tableGlobeViewObject : Transform
- tableGlobeCameraPivotObject : Transform
- tableGlobePivotObject : GameObject
- tableGlobeCountryMarkerObject : GameObject
- tableGlobeCountryMarkerObjectMaterials : List<Renderer>
- tableGlobeCompleteSounds : AudioSource[]
- musicBoxBooleanStates : bool[]
- musicBoxPictureFalling : bool = false
- musicBoxPictureSoundPlayed : bool = false
- musicBoxPictureSeen : boo = false
- musicBoxItemBooleanStates : bool[]
- musicBoxStandViewExitTimer : float = 0.0
- CarouselMusicBoxScreenMaterialSet : bool = false
- CarouselMusicBoxScreenMaterialSwitchIntervalTimer : float = 0.0
- CarouselMusicBoxScreenMaterialsSwitched : int = 0
- CarouselMusicBoxScreensPaused : bool = false
- CarouselMusicBoxChimePlayed ; bool = false
- CarouselMusicBoxExitTimer : float = 0.0
- CarouselMusicBoxExitAnimationActive : bool = false
- musicBoxPuzzleBooleanStates : bool []
- musicBoxPuzzleProgress : MusicBoxPuzzleProgressStates
- musicBoxColliderObjectViewActive : bool = false
- musicBoxViewObjectViewReset ; bool = false
- musicBoxItemRotateSpeed : float = 0.0
- musicBoxItemReset : bool[]
- musicBoxWindUpDiscTimer : float = 0.0
- musicBoxAssembled : bool = false
- musicBoxChimePlayed : bool = false
- musicBoxExitTimer : bool = false
- musicBoxLidMirrorCameraAtPosition : bool = false
- musicBoxHideSequenceActive : bool = false
- musicBoxHideSoundPlayed : bool = false
- musicBoxHideBooleanStates : bool[]
- musicBoxHideTimer : float = 0.0
- musicBoxPictureViewObject : Transform
- musicBoxPictureObject : GameObject
- musicBoxPictureAnimator : Animator
<<enumeration>>
PuzzleInteraction
SafePuzzle
TablePiecePuzzle
MusicBoxPuzzle
PianoPuzzle
<<enumeration>>
SafeStates
SafeInteractable
SafeViewActive
SafeViewExit
SafeDialOperational
SafeOpen
SafeReset
<<enumeration>>
SafeDialStates
None
Active FirstLeft
SecondRight
ThirdLeft
Open
<<enumeration>>
TablePieceStates
ObtainingTableKey
TablePieceInteractable
TablePieceViewActive
TablePieceViewSet
TableKeyPlaceable
InsertingTableKey
TableKeyInserted
<<enumeration>>
TableGlobeStates
TableGlobeInteractable
TableGlobeViewActive
TableGlobeViewSet
TableGlobeCameraAtPosition
<<enumeration>>
TableGlobePuzzleProgressStates
None
FirstIteration
SecondIteration
ThirdIteration
Complete
<<enumeration>>
TableGlobeMarkerLights
UK
India
Europe
Africa
SouthUSA
NorthUSA
Canada
Australia
China
Russia
Greenland
<<enumeration>>
TableGlobePuzzleStates
TableGlobePuzzleBegin
TableGlobePuzzleSetProgress
TableGlobeRotateable
TableGlobePuzzleComplete
<<enumeration>>
MusicBoxStates
MusicBoxPictureViewActive
MusicBoxPictureInteractable
MusicBoxAndItemsObtainable
MusicBoxItemsObtained
MusicBoxStandInteractable
MusicBoxAndItemsObtained
MusicBoxStandViewSet
MusicBoxStandViewActive
MusicBoxPlaceable
PlacingMusicBox
MusicBoxPlaced
MusicBoxInteractable
MusicBoxViewActive
MusicBoxCameraAtPosition
MusicBoxViewSet
PlayerController
InteractionController
ButtonController
SafePuzzleLightAnimation
+ safePuzzleLightAnimateState : string = ""
+ SetSafeLightOn() : void
+ SetSafeLightOff() : void
GlobePuzzleLightAnimation
+ globePuzzleLightAnimateState : string = ""
+ SetGlobeLightOn() : void
+ SetGlobeLightOff() : void
GlobePuzzleGlobeAnimation
+ globePuzzleAnimationState : string = ""
+ SetAnimationHoverEnd() : void
+ SetAnimationOneEnd() : void
+ SetAnimationTwoEnd() : void
+ SetAnimationThreeEnd() : void
+ SetAnimationUKLightOn() : void
+ SetAnimationUKLightOff() : void
+ SetAnimationIndiaLightOn() : void
+ SetAnimationIndiaLightOff() : void
+ SetAnimationEuropeLightOn() : void
+ SetAnimationEuropeLightOff() : void
+ SetAnimationAfricaLightOn() : void
+ SetAnimationAfricaLightOff() : void
+ SetAnimationSouthUSALightOn() : void
+ SetAnimationSouthUSALightOff() : void
+ SetAnimationNorthUSALightOn() : void
+ SetAnimationNorthUSALightOff() : void
+ SetAnimationCanadaLightOn() : void
+ SetAnimationCanadaLightOff() : void
+ SetAnimationAustraliaLightOn() : void
+ SetAnimationAustraliaLightOff() : void
+ SetAnimationChinaLightOn() : void
+ SetAnimationChinaLightOff() : void
+ SetAnimationRussiaLightOn() : void
+ SetAnimationRussiaLightOff() : void
+ SetAnimationGreenlandLightOn() : void
+ SetAnimationGreenlandLightOff() : void
+ GlobeAnimationReset() : void
MusicBoxPuzzlePictureAnimation
+ musicBoxPuzzlePictureAnimationState : string = ""
+ SetAnimationSwingingEnd() : void
+ SetAnimationFallingImpact() : void
+ SetAnimationFallingEnd() : void
+ MusicBoxPuzzlePictureAnimationReset() : void
MusicBoxPuzzlePictureAnimation
+ MusicBoxPuzzlePictureLightAnimation : string = ""
+ SetPictureLightOn() : void
+ SetPictureLightOff() : void
MusicBoxPuzzleLightAnimation
+ musicBoxPuzzleLightAnimateState : string = ""
+ SetMusicBoxStandLightOn() : void
+ SetMusicBoxStandLightOff() : void
MusicBoxPuzzleCarouselAnimation
+ musicBoxPuzzleCarouselAnimateState : string = ""
+ SetCarouselShake() : void
+ SetCarouselButtonPressedIn() : void
+ SetCarouselDrawerOut() : void
+ CarouselAnimationReset() : void
MusicBoxPuzzleWardrobeDoorAnimation
+ musicBoxPuzzleWardrobeDoorAnimationState : string = ""
+ SetMusicBoxWardrobeDoorsOpen() : void
+ SetMusicBoxWardrobeDoorsClosed() : void
+ SetMusicBoxWardrobeDoorsOpenSlightly() : void
+ MusicBoxPuzzleWardrobeDoorAnimationReset() : void
PianoPuzzleBookStandLightAnimation
+ pianoPuzzleBookStandLightAnimateState : string = ""
+ SetBookStandLightOn() : void
+ SetBookStandLightOff() : void

*Figure 64: 'PuzzleRoomTwo' class, part one of three illustrations of the class's substantial level of association and instances of enumeration*



```
<<enumeration>>
MusicBoxItemStates

ObtainingMusicBoxLatchKey
MusicBoxLatchKeyObtained
ObtainingMusicBoxWindUpKey
MusicBoxWindUpKeyObtained
CarouselButtonViewActive
CarouselButtonInteractable
CarouselCylinderViewActive
CarouselCylinderCameraAtPosition
CarouselComplete
ObtainingMusicBoxDisc
MusicBoxDiscObtained
ObtainingMusicBox
MusicBoxObtained
```

```
<<enumeration>>
MusicBoxPuzzleStates

LatchKeyAligned
LatchKeyInserted
LatchKeyComplete
LidHingeProppedOpen
LidHingeComplete
WindUpKeyAligned
WindUpKeyInserted
WindUpKeyComplete
DiscComplete
DiscNeedleComplete
```

```
<<enumeration>>
MusicBoxPuzzleProgressStates

None
LatchKeyCurrent
LidHingeCurrent
WindUpKeyCurrent
DiscCurrent
DiscNeedleCurrent
```

```
<<enumeration>>
MusicBoxItems

MusicBoxLatchKey
MusicBoxWindUpKey
MusicBoxDisc
```

```
<<enumeration>>
MusicBoxHideStates

WardrobeDoorsOpening
WardrobeDoorsOpened
WardrobeDoorsClosing
PlayerHiding
WardrobeExit
PlayerEmergeSlightly
PlayerEmergeFully
```

```
<<enumeration>>
MusicBoxAndItemsDummy

MusicBoxDummy
MusicBoxLatchKeyDummy
MusicBoxWindUpKeyDummy
MusicBoxDiscDummy
```

```
<<enumeration>>
MusicBoxColliders

MusicBox
MusicBoxBase
MusicBoxLid
MusicBoxLatchTop
MusicBoxLatchKeyHole
MusicBoxWindUpKeyHole
MusicBoxDiscSpindle
MusicBoxDiscNeedle
```

```
<<enumeration>>
MusicBoxItemColliderViewObjects

MusicBoxWindUpKeyView
MusicBoxLatchKeyView
MusicBoxLidHingeView
MusicBoxDiscView
MusicBoxDiscNeedleView
MusicBoxWindUpKeyFullView
MusicBoxLidMirrorView
```

```
<<enumeration>>
MusicBoxAndItems

MusicBox
MusicBoxLatchKey
MusicBoxWindUpKey
MusicBoxDisc
```

```
<<enumeration>>
MusicBoxParts

MusicBoxLidHinge
MusicBoxDiscNeedle
```

```
<<enumeration>>
MusicBoxItemPositions

LatchKeyReset
LatchKeyHole
LatchKeyInsert
LidHingeProppedOpenReset
LidHingeOpen
WindUpKeyReset
WindUpKeyHole
WindUpKeyInsert
DiscReset
DiscPlaced
DiscNeedleReset
DiscNeedleAligned
```

```
- musicBoxPictureSounds : AudioSource[]
- musicBoxPictureLightSound : AudioSource
- musicBoxPictureLightAnimator : Animator
- musicBoxAndItemsDummy : GameObject[]
- musicBoxLatchKeyDummyAnimator : Animator
- musicBoxStandObject : GameObject
- musicBoxStandViewObject : Transform
- musicBoxStandCameraPivotObject : Transform
- musicBoxStandLightSound : AudioSource
- musicBoxStandLightBulbObject : GameObject
- musicBoxStandLightBulbRenderer : Renderer
- musicBoxStandLightAnimator : Animator
- musicBoxStandLightPlaneAnimator : Animator
- CarouselMusicBoxObject : GameObject
- CarouselMusicBoxAnimator : Animator
- CarouselMusicBoxButtonViewObject : Transform
- CarouselMusicBoxCylinderViewObject : Transform
- CarouselMusicBoxScreenObject : GameObject
- CarouselMusicBoxScreenRenderers : List<Renderer>
- CarouselMusicBoxScreenVideoPlayers : List<VideoPlayer>
- CarouselMusicBoxScreenMaterials : Material[]
- CarouselMusicBoxSounds : AudioSource[]
- CarouselMusicBoxEyesObject : GameObject
- CarouselMusicBoxEyesAnimator : Animator
- CarouselMusicBoxCandleAnimators : Animator[]
- musicBoxViewObject : Transform
- musicBoxCameraPivotObject : Transform
- musicBoxColliderObjects : Collider[]
- musicBoxColliderCameraPivotObject : GameObject
- musicBoxItemColliderViewObjects : List<Transform>
- musicBoxAndItems : GameObject[]
- musicBoxParts : GameObject[]
- musicBoxItemPositionsObject : GameObject
- musicBoxItemPositions : List<Transform>
- musicBoxItemSounds : AudioSource[]
- musicBoxAnimator : Animator
- musicBoxChime : AudioSource
- musicBoxHandPrintObject : GameObject
- musicBoxHideSounds : AudioSource[]
- musicBoxHandPrintAnimator : Animator
- musicBoxWardrobeObject : GameObject
- musicBoxWardrobeAnimator : Animator
- musicBoxWardrobeSounds : AudioSource[]
- bookshelfLightPoweredOn : bool = false
- bookshelfLightToggled : bool = false
- musicSheetBookStates : bool[]
- pianoPuzzleBooleanStates : bool[]
- musicSheetBookCurrentPage : MusicSheetBookCurrentPage
- pianoPuzzleBooleanStates : bool[]
- playMusicSheetBookPageTurnSound : bool = false
- playMusicSheetBookCoverImpactSound : bool = false
- pianoShelfViewExitTimer : float = 0.0
- pianoKeyViewActive : boo = true
- pianoKeyNoteInput : List<string>
- pianoKeyPlayed : bool = false
- pianoKeyPlayedWaitCount : int[]
- pianoKeyNoteCorrectSequence : string[] = { "G", "D", "G", "A", "D", "A", "A#", "A", "G", "A", "D" }
- pianoKeyNotesPlayedCorrectCount : int = 0
- pianoKeyNotePlayedTimer : float = 0.0
- pianoKeyNotePlayedResetCount : int = 0
- pianoKeyNoteReset : bool = false
- pianoCompleteChimePlayed : bool = false
- pianoAnimator : Animator
- bookshelfLightAnimator : Animator
- bookshelfLightSwitchObject : GameObject
- bookshelfLightSwitchAnimator : Animator
- bookshelfLightSwitchSounds : AudioSource[]
- musicSheetBookDummyObject : GameObject
- musicSheetBookViewObject : Transform
- bookStandLightBulbObject : GameObject
- bookStandLightBulbRenderer : Renderer
- bookStandLightAnimator : Animator
- bookStandObject : GameObject
- bookStandLightSound : AudioSource
- bookStandViewObject : Transform
- musicSheetBookObject : GameObject
- placeMusicSheetBookSounds : AudioSource[]
- musicSheetBookAnimator : Animator
- musicSheetDummyObject : GameObject
- strobeLightAnimator : Animator
- strobeLightSound : AudioSource
- pianoViewObjects : Transform[]
- musicSheetObject : GameObject
- placeMusicSheetSound : AudioSource
- pianoKeyInteractionObjectCollider : Collider
- pianoKeyObject : GameObject
- pianoKeyAnimators : List<Animator>
- pianoSounds : AudioSource[]

- Start() : void
- Update() : void
# RayHitPressed() : void
# RayHtHeld() : void
# ObjectAtTransform(objectTransform : Transform, targetTransform : Transform, objectAtPosition : bool &) : void
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion,
objectViewActive : bool &) : bool
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion) : bool
# GameObjectAtPreviousTransform(gameObjectTransform : GameObject, previousGameObjectPosiiton : Vector3,
previousGameObjectRotation : Quaternion, gameObjectTransformEqual : bool &, desiredState : bool) : void
# StorePreviousTransform(previousCameraTransform : Transform, previousCameraPosition : Vector3 &, previousCameraRotation :
Quaternion &) : void
# GameObjectObtained(puzzleItem : GameObject) : bool
- SafePictureMove() : void
- EnterSafeView() : void
- ExitSafeView() : void
- SafeDialRotation() : void
- SafeReset() : void
- SafeCracking() : void
- EnterTablePieceView() : void
- ExitTablePieceView() : void
- PlaceTablePieceKey() : void
- EnterTableGlobeView() : void
- ExitTableGlobeView() : void
- ResetGlobePuzzle() : void
- CheckTableGlobeCountryOrder() : void
- IterateTableGlobePuzzle() : void
- TablePieceSolving() : void
- MusicBoxPictureAnimate() : void
- EnterMusicBoxPictureView() : void
- ExitMusicBoxPictureView() : void
- MusicBoxSolving() : void
- ObtainMusicBoxItems() : void
- CarouselMusicBoxAnimate() : void
- EnterCarouselButtonView() : void
- EnterCarouselCylinderView() : void
- ExitCarouselCylinderView() : void
- EnterMusicBoxStandView() : void
- ExitMusicBoxStandView() : void
- PlaceMusicBox() : void
- EnterMusicBoxView() : void
- ExitMusicBoxView() : void
- MusicBoxAssembling() : void
- MusicBoxCameraControl() : void
- MusicBoxHide() : void
- PianoPuzzleSolving() : void
- ToggleBookshelfLight() : void
- EnterPianoMusicSheetBookView() : void
- ExitPianoMusicSheetBookView() : void
- ObtainPianoMusicSheetBook() : void
- EnterPianoBookStandView() : void
- ExitPianoBookStandView() : void
- PlaceMusicSheetBook() : void
- MusicSheetBookFlicker() : void
- ObtainPianoMusicSheet() : void
- EnterPianoView() : void
- ExitPianoView() : void
- EnterPianoShelfView() : void
- ExitPianoShelfView() : void
- PlaceMusicSheet() : void
- EnterPianoKeyView() : void
- AlternatePianoCamera() : void
- CheckPianoNotesPlayed() : void
```

```
PianoPuzzleBookPageTurnAnimation

+ pianoPuzzleBookPageTurnAnimateState : string = ""

+ SetBookPageTurn() : void
+ SetBookPageTurned() : void
```

```
PianoPuzzleStrobeLightAnimation

+ pianoPuzzleStrobeLightAnimateState : string = ""

+ SetStrobeLightOn() : void
+ PianoPuzzleStrobeLightAnimationReset() : void
```

```
<<enumeration>>
PianoSounds

CompleteChime
ErrorChime
KeyC
KeyCs
KeyD
KeyDs
KeyE
KeyF
KeyFs
KeyG
KeyGs
KeyA
KeyAs
KeyB
```

*Figure 66: 'PuzzleRoomTwo' class, part two of three illustrations of the class's substantial level of association and instances of enumeration*

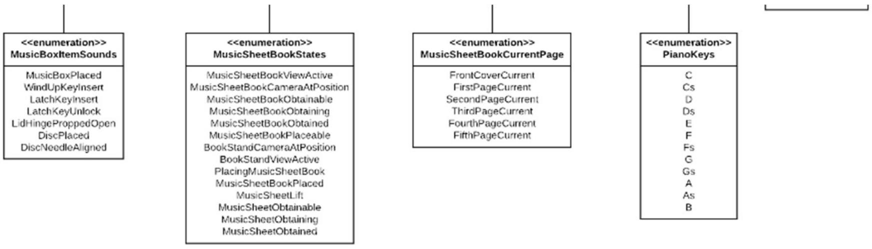


*Figure 65: 'PuzzleRoomTwo' class, part three of three illustrations of the class's substantial level of association and instances of enumeration*

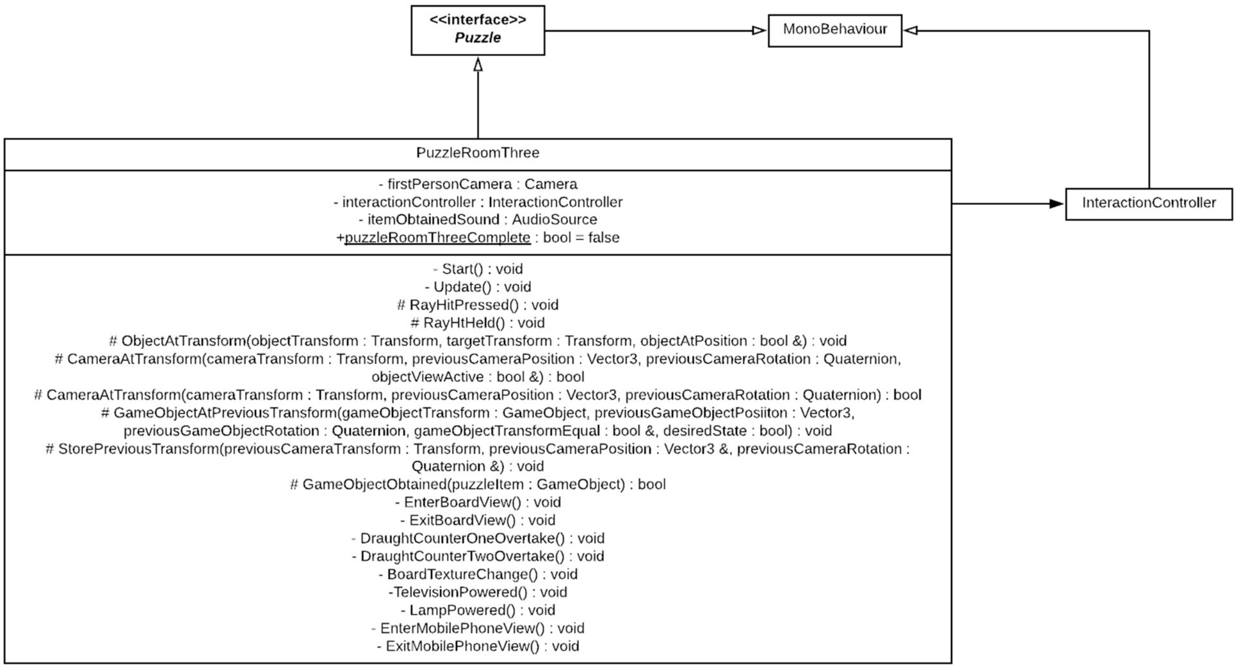

<<interface>>
Puzzle
MonoBehaviour
PuzzleRoomThree
- firstPersonCamera : Camera
- interactionController : InteractionController
- itemObtainedSound : AudioSource
+puzzleRoomThreeComplete : bool = false
- Start() : void
- Update() : void
# RayHitPressed() : void
# RayHtHeld() : void
# ObjectAtTransform(objectTransform : Transform, targetTransform : Transform, objectAtPosition : bool &) : void
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion, objectViewActive : bool &) : bool
# CameraAtTransform(cameraTransform : Transform, previousCameraPosition : Vector3, previousCameraRotation : Quaternion) : bool
# GameObjectAtPreviousTransform(gameObjectTransform : GameObject, previousGameObjectPosiiton : Vector3, previousGameObjectRotation : Quaternion, gameObjectTransformEqual : bool &, desiredState : bool) : void
# StorePreviousTransform(previousCameraTransform : Transform, previousCameraPosition : Vector3 &, previousCameraRotation : Quaternion &) : void
# GameObjectObtained(puzzleItem : GameObject) : bool
- EnterBoardView() : void
- ExitBoardView() : void
- DraughtCounterOneOvertake() : void
- DraughtCounterTwoOvertake() : void
- BoardTextureChange() : void
-TelevisionPowered() : void
- LampPowered() : void
- EnterMobilePhoneView() : void
- ExitMobilePhoneView() : void
InteractionController

*Figure 67: 'PuzzleRoomThree' class, identify the only association, to the 'InteractionController' class (classes functionality is conceptual and not developed)*

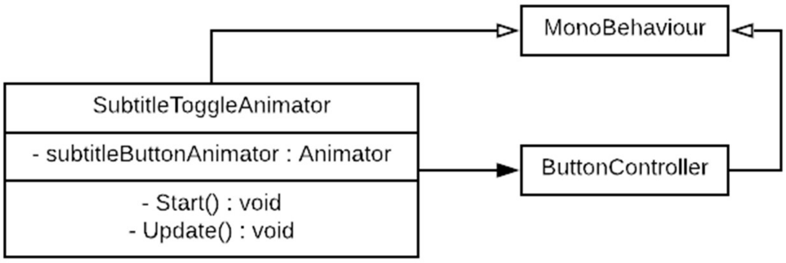


*Figure 68: 'SubtitleToggleAnimator' class, showing the only association, to the 'ButtonController' class*

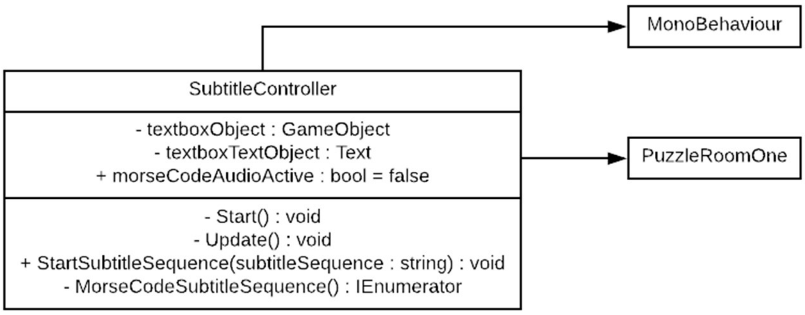

MonoBehaviour
SubtitleController
- textboxObject : GameObject
- textboxTextObject : Text
+ morseCodeAudioActive : bool = false
- Start() : void
- Update() : void
+ StartSubtitleSequence(subtitleSequence : string) : void
- MorseCodeSubtitleSequence() : IEnumerator
PuzzleRoomOne

*Figure 69: 'SubtitleController' class, illustrating the only association, to the 'PuzzleRoomOne' class*

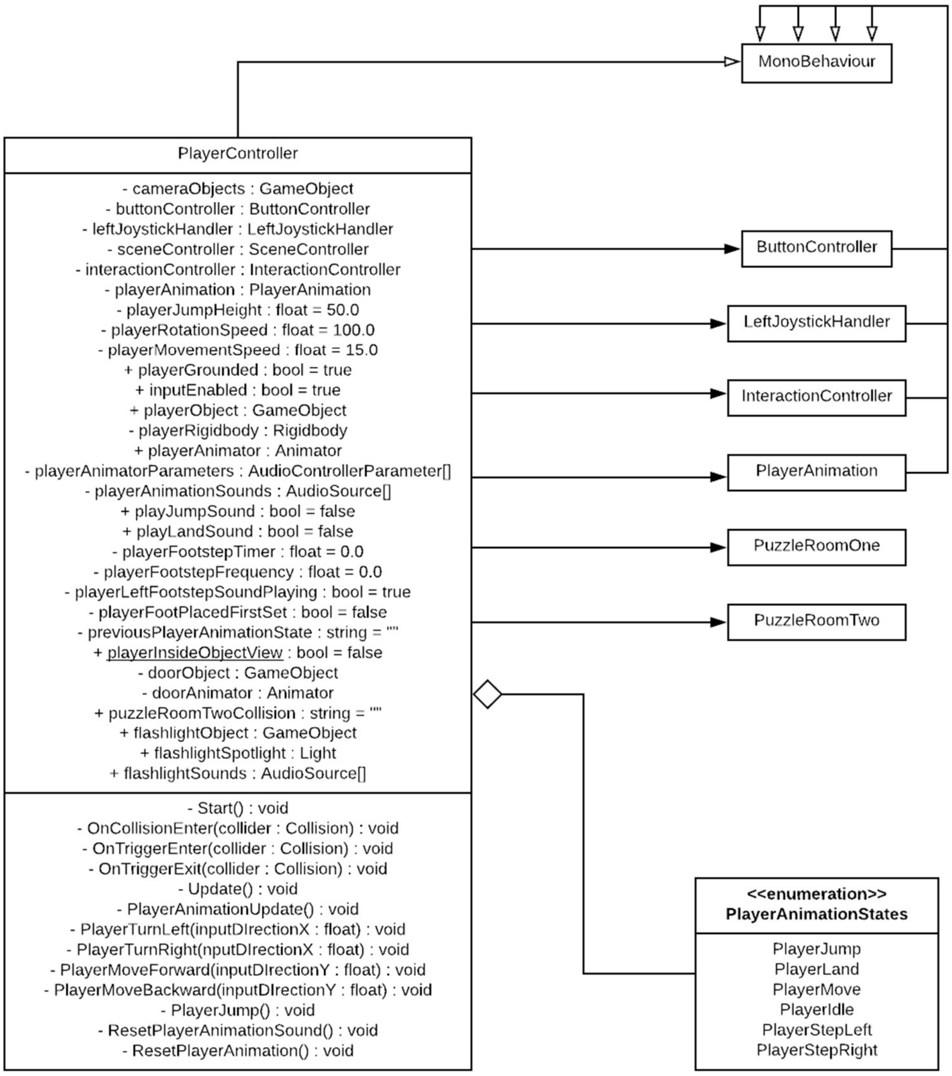

MonoBehaviour
PlayerController
- cameraObjects : GameObject
- buttonController : ButtonController
- leftJoystickHandler : LeftJoystickHandler
- sceneController : SceneController
- interactionController : InteractionController
- playerAnimation : PlayerAnimation
- playerJumpHeight : float = 50.0
- playerRotationSpeed : float = 100.0
- playerMovementSpeed : float = 15.0
+ playerGrounded : bool = true
+ inputEnabled : bool = true
+ playerObject : GameObject
- playerRigidbody : Rigidbody
+ playerAnimator : Animator
- playerAnimatorParameters : AudioControllerParameter[]
- playerAnimationSounds : AudioSource[]
+ playJumpSound : bool = false
+ playLandSound : bool = false
- playerFootstepTimer : float = 0.0
- playerFootstepFrequency : float = 0.0
- playerLeftFootstepSoundPlaying : bool = true
- playerFootPlacedFirstSet : bool = false
- previousPlayerAnimationState : string = ""
+ playerInsideObjectView : bool = false
- doorObject : GameObject
- doorAnimator : Animator
+ puzzleRoomTwoCollision : string = ""
+ flashlightObject : GameObject
+ flashlightSpotlight : Light
+ flashlightSounds : AudioSource[]
- Start() : void
- OnCollisionEnter(collider : Collision) : void
- OnTriggerEnter(collider : Collision) : void
- OnTriggerExit(collider : Collision) : void
- Update() : void
- PlayerAnimationUpdate() : void
- PlayerTurnLeft(inputDIrectionX : float) : void
- PlayerTurnRight(nputDIrectionX : float) : void
- PlayerMoveForward(inputDIrectionY : float) : void
- PlayerMoveBackward(inputDIrectionY : float) : void
- PlayerJump() : void
- ResetPlayerAnimationSound() : void
- ResetPlayerAnimation() : void
ButtonController
LeftJoystickHandler
InteractionController
PlayerAnimation
PuzzleRoomOne
PuzzleRoomTwo
<<enumeration>>
PlayerAnimationStates
PlayerJump
PlayerLand
PlayerMove
PlayerIdle
PlayerStepLeft
PlayerStepRight

*Figure 70: 'PlayerController' class, identify the associations to the 'ButtonController', 'LeftJoystickHandler', 'InteractionController', 'PlayerAnimation' and puzzle room classes, as well as the instance of enumeration*

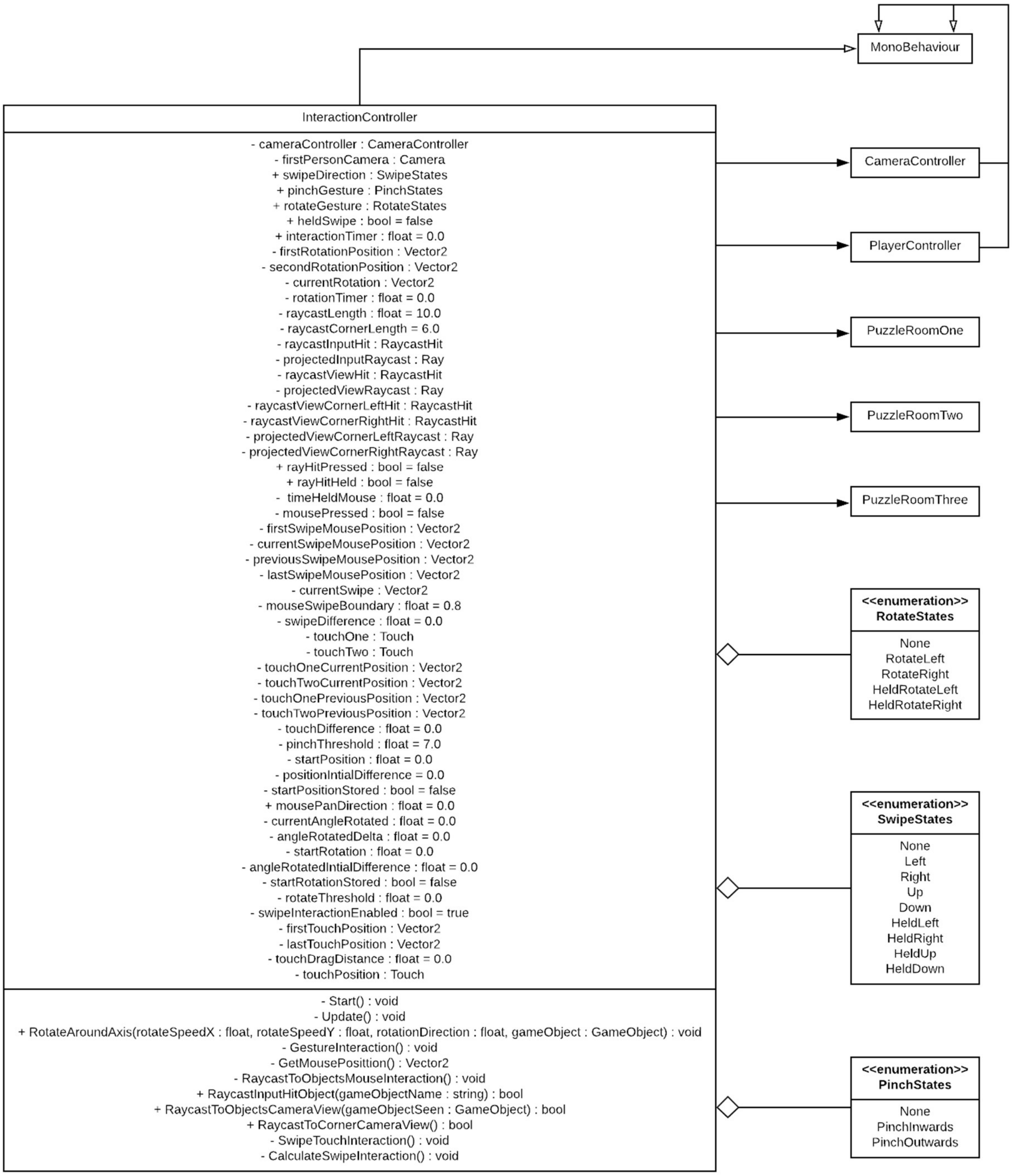

MonoBehaviour
InteractionController
- cameraController : CameraController
- firstPersonCamera : Camera
+ swipeDirection : SwipeStates
+ pinchGesture : PinchStates
+ rotateGesture : RotateStates
+ heldSwipe : bool = false
+ interactionTimer : float = 0.0
- firstRotationPosition : Vector2
- secondRotationPosition : Vector2
- currentRotation : Vector2
- rotationTimer : float = 0.0
- raycastLength : float = 10.0
- raycastCornerLength = 6.0
- raycastInputHit : RaycastHit
- projectedInputRaycast : Ray
- raycastViewHit : RaycastHit
- projectedViewRaycast : Ray
- raycastViewCornerLeftHit : RaycastHit
- raycastViewCornerRightHit : RaycastHit
- projectedViewCornerLeftRaycast : Ray
- projectedViewCornerRightRaycast : Ray
+ rayHitPressed : bool = false
+ rayHitHeld : bool = false
- timeHeldMouse : float = 0.0
- mousePressed : bool = false
- firstSwipeMousePosition : Vector2
- currentSwipeMousePosition : Vector2
- previousSwipeMousePosition : Vector2
- lastSwipeMousePosition : Vector2
- currentSwipe : Vector2
- mouseSwipeBoundary : float = 0.8
- swipeDifference : float = 0.0
- touchOne : Touch
- touchTwo : Touch
- touchOneCurrentPosition : Vector2
- touchTwoCurrentPosition : Vector2
- touchOnePreviousPosition : Vector2
- touchTwoPreviousPosition : Vector2
- touchDifference : float = 0.0
- pinchThreshold : float = 7.0
- startPosition : float = 0.0
- positionIntialDifference = 0.0
- startPositionStored : bool = false
+ mousePanDirection : float = 0.0
- currentAngleRotated : float = 0.0
- angleRotatedDelta : float = 0.0
- startRotation : float = 0.0
- angleRotatedIntialDifference : float = 0.0
- startRotationStored : bool = false
- rotateThreshold : float = 0.0
- swipeInteractionEnabled : bool = true
- firstTouchPosition : Vector2
- lastTouchPosition : Vector2
- touchDragDistance : float = 0.0
- touchPosition : Touch
- Start() : void
- Update() : void
+ RotateAroundAxis(rotateSpeedX : float, rotateSpeedY : float, rotationDirection : float, gameObject : GameObject) : void
- GestureInteraction() : void
- GetMousePositiion() : Vector2
- RaycastToObjectsMouseInteraction() : void
+ RaycastInputHitObject(gameObjectName : string) : bool
+ RaycastToObjectsCameraView(gameObjectSeen : GameObject) : bool
+ RaycastToCornerCameraView() : bool
- SwipeTouchInteraction() : void
- CalculateSwipeInteraction() : void
CameraController
PlayerController
PuzzleRoomOne
PuzzleRoomTwo
PuzzleRoomThree
<<enumeration>>
RotateStates
None
RotateLeft
RotateRight
HeldRotateLeft
HeldRotateRight
<<enumeration>>
SwipeStates
None
Left
Right
Up
Down
HeldLeft
HeldRight
HeldUp
HeldDown
<<enumeration>>
PinchStates
None
PinchInwards
PinchOutwards

*Figure 71: 'InteractionController' class, showcase the multiple instances of enumeration and association to the 'CameraController', 'PlayerController' and puzzle room classes*

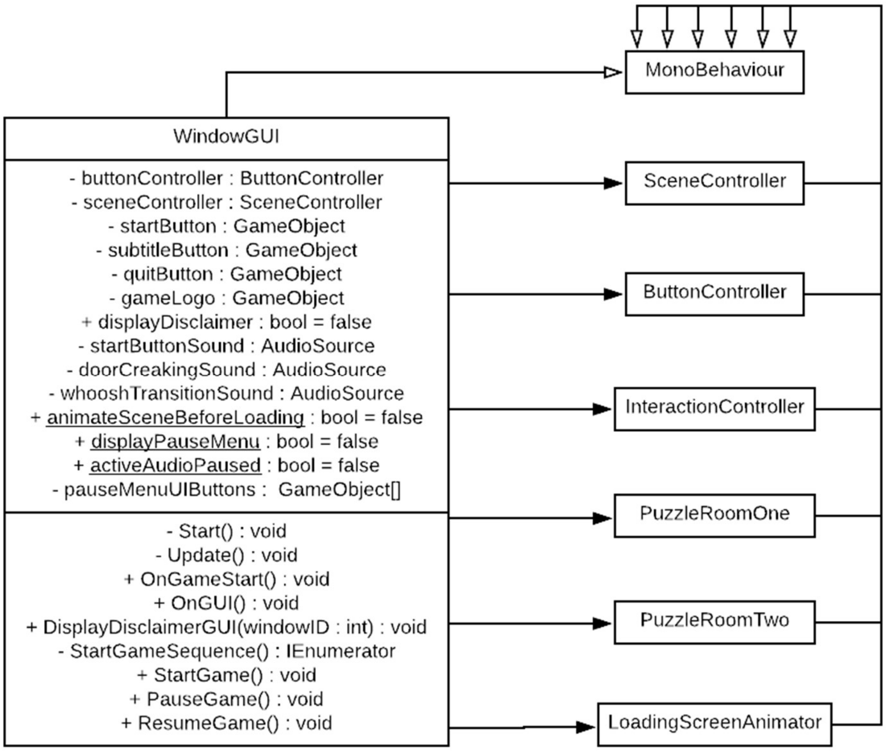

MonoBehaviour
WindowGUI
- buttonController : ButtonController
- sceneController : SceneController
- startButton : GameObject
- subtitleButton : GameObject
- quitButton : GameObject
- gameLogo : GameObject
+ displayDisclaimer : bool = false
- startButtonSound : AudioSource
- doorCreakingSound : AudioSource
- whooshTransitionSound : AudioSource
+ animateSceneBeforeLoading : bool = false
+ displayPauseMenu : bool = false
+ activeAudioPaused : bool = false
- pauseMenuUIButtons : GameObject[]
- Start() : void
- Update() : void
+ OnGameStart() : void
+ OnGUI() : void
+ DisplayDisclaimerGUI(windowID : int) : void
- StartGameSequence() : IEnumerator
+ StartGame() : void
+ PauseGame() : void
+ ResumeGame() : void
SceneController
ButtonController
InteractionController
PuzzleRoomOne
PuzzleRoomTwo
LoadingScreenAnimator

*Figure 72: 'WindowGUI' class, showcase the multiple associations to the 'SceneController', 'ButtonController', 'InteractionController', 'LoadingScreenAnimator' and puzzle room classes*

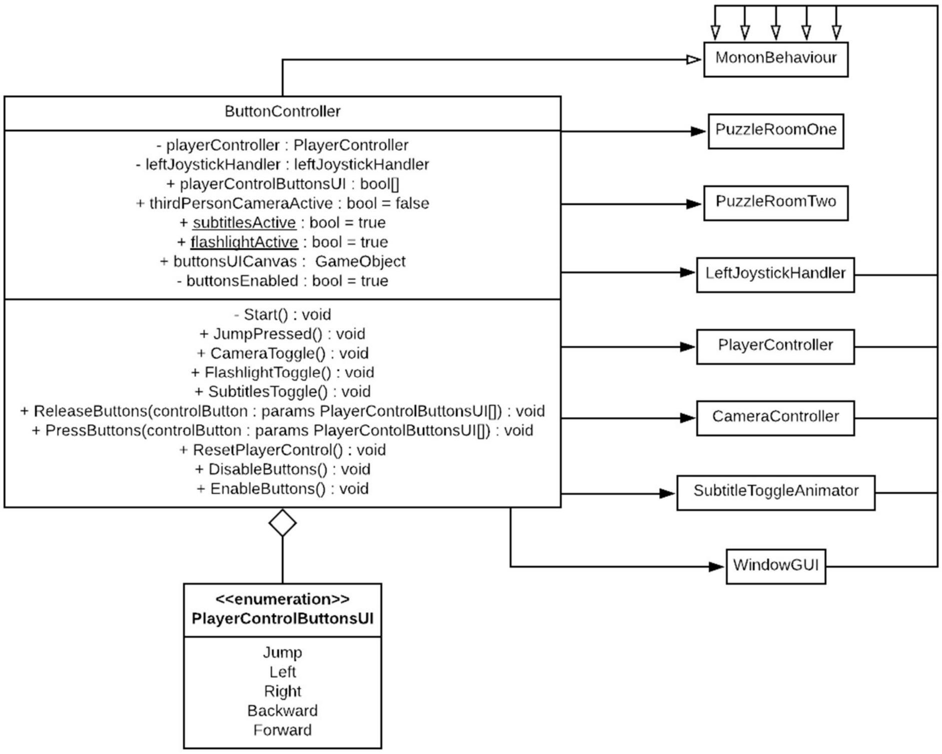

MononBehaviour
ButtonController
- playerController : PlayerController
- leftJoystickHandler : leftJoystickHandler
+ playerControlButtonsUI : bool[]
+ thirdPersonCameraActive : bool = false
+ subtitlesActive : bool = true
+ flashlightActive : bool = true
+ buttonsUICanvas : GameObject
- buttonsEnabled : bool = true
- Start() : void
+ JumpPressed() : void
+ CameraToggle() : void
+ FlashlightToggle() : void
+ SubtitlesToggle() : void
+ ReleaseButtons(controlButton : params PlayerControlButtonsUI[]) : void
+ PressButtons(controlButton : params PlayerContolButtonsUI[]) : void
+ ResetPlayerControl() : void
+ DisableButtons() : void
+ EnableButtons() : void
<<enumeration>>
PlayerControlButtonsUI
Jump
Left
Right
Backward
Forward
PuzzleRoomOne
PuzzleRoomTwo
LeftJoystickHandler
PlayerController
CameraController
SubtitleToggleAnimator
WindowGUI

*Figure 73: 'ButtonController' class, identify the association to the 'LeftJoystickHandler', 'PlayerController', 'CameraController', 'WindowGUI', 'SubtitleToggleAnimator' and puzzle room classes*

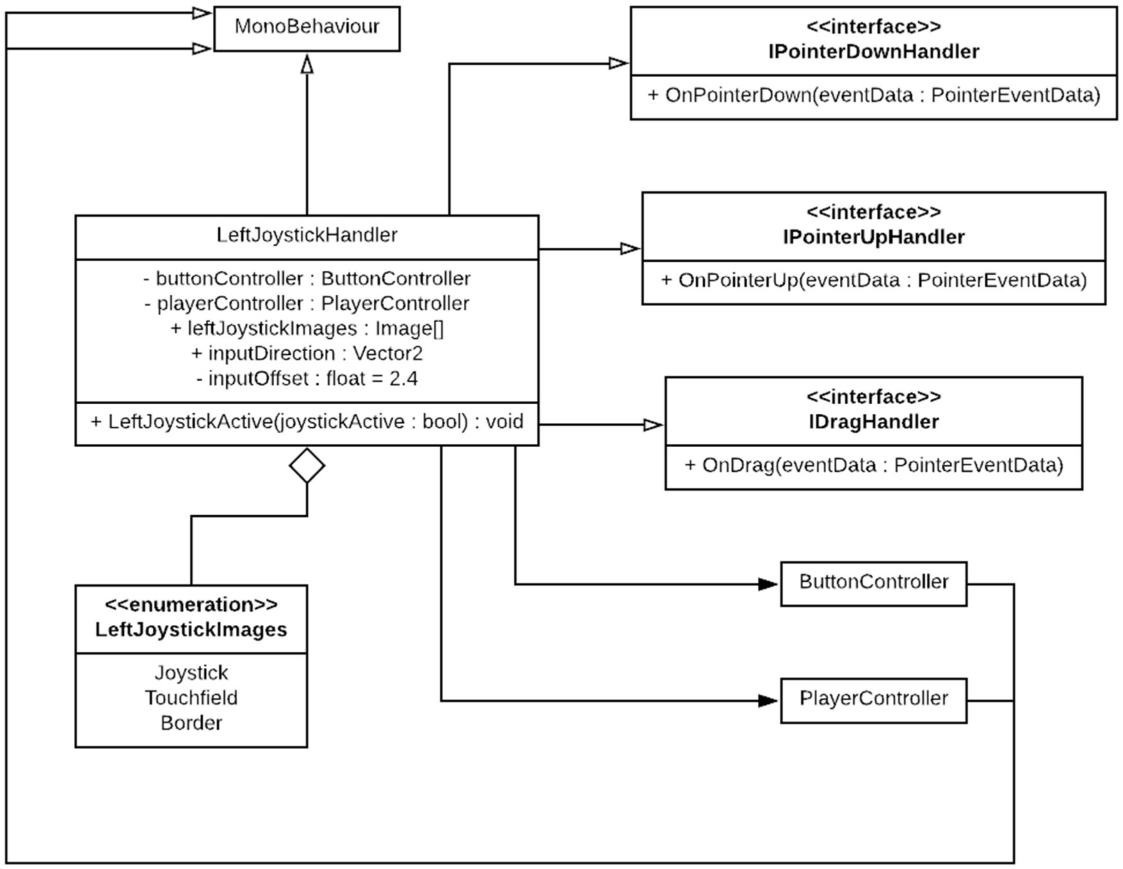

MonoBehaviour
<<interface>>
IPointerDownHandler
+ OnPointerDown(eventData : PointerEventData)
LeftJoystickHandler
- buttonController : ButtonController
- playerController : PlayerController
+ leftJoystickImages : Image[]
+ inputDirection : Vector2
- inputOffset : float = 2.4
+ LeftJoystickActive(joystickActive : bool) : void
<<interface>>
IPointerUpHandler
+ OnPointerUp(eventData : PointerEventData)
<<interface>>
IDragHandler
+ OnDrag(eventData : PointerEventData)
<<enumeration>>
LeftJoystickImages
Joystick
Touchfield
Border
ButtonController
PlayerController

*Figure 74: 'LeftJoystickHandler' class, display the association to the 'ButtonController' and 'PlayerController' classes, as well as the application of enumeration and multiple instances of inheritance*

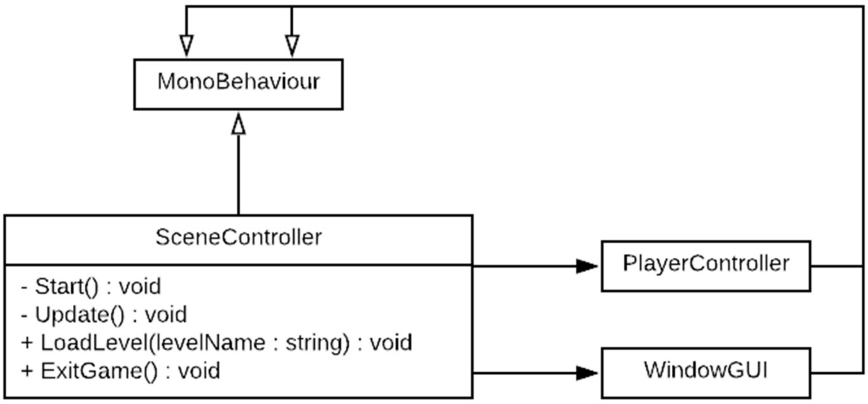


*Figure 75: 'SceneController' class, showcasing the association to the 'PlayerController' and 'WindowGUI' classes*

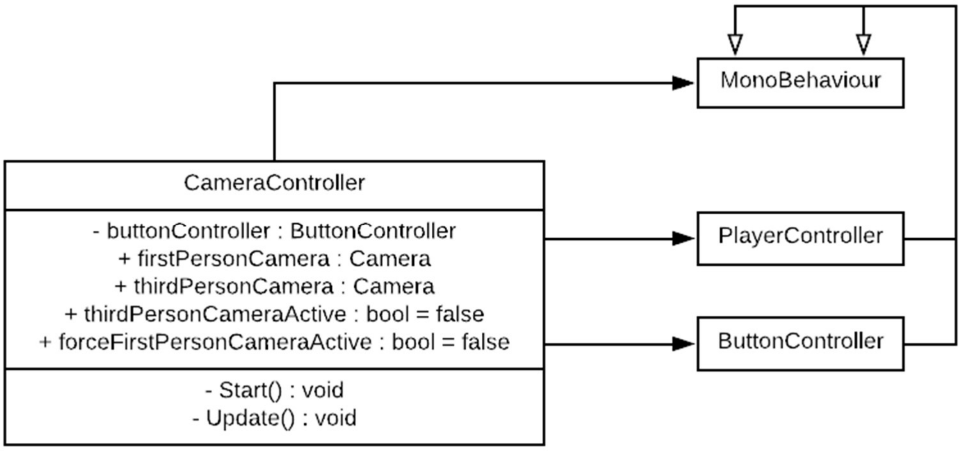

MonoBehaviour
CameraController
- buttonController : ButtonController
+ firstPersonCamera : Camera
+ thirdPersonCamera : Camera
+ thirdPersonCameraActive : bool = false
+ forceFirstPersonCameraActive : bool = false
- Start() : void
- Update() : void
PlayerController
ButtonController

*Figure 76: 'CameraController' class, illustrate the association to the 'PlayerController' and 'ButtonController' classes*

***Appendix I***:

Tabularised beneath this passage, details the safe puzzle implementation, as presented in the second puzzle room scene. A step-by-step process for the puzzles continuity is presented, alongside the mechanisms that address the scenarios revealed; all of the mechanisms that are detailed in the following table, reside within the 'InteractionController' and 'PuzzleRoomTwo' classes (see ***Appendix H***).

*Table 1: Puzzle room two, safe puzzle mechanical implementation, detailing the functional invocation of the puzzle's components*

| ***Safe puzzle: mechanical implementation*** | | | |
|---|---|---|---|
| ***Step*** | ***Process*** | ***Mechanisms employed*** | ***How have the mechanisms been addressed?*** |
| 1 | Activate the movement of the picture canvas object, to enable the safe object interaction. | Touch-based interaction with the picture canvas object, to invoke the 'X' dimensional translation of the picture canvas objects position and accompanying audio cues. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the picture canvas object is pressed within any screen point occupied by the application. This invokes the 'SafePictureMove' method within the 'PuzzleRoomTwo' script. |
| 2 | Crack the safe to progress to the table piece puzzle. | Touch-based interaction with the safe door or safe dial objects, to enter the safe interaction view, via the first-person camera. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the safe door or safe dial objects are pressed within any screen point occupied by the application. This invokes the 'EnterSafeView' method within the 'PuzzleRoomTwo' script. |
| | | Swipe gesture-based interaction when within the safe interaction view, to invoke the rotary movements of the safe dial object and accompanying audio cues. | Swipe gesture-based interaction is governed by the 'InteractionController' class, within the 'GestureInteraction' method. Upon a held leftwards or swipe left interaction being performed, the 'swipeDIrection' enumerator object is set to 'HeldLeft' of 'Left'. This is used to invoke the 'SafeDialRotation' method within the 'PuzzleRoomTwo' script; which rotates the safe dial object anti-clockwise for every frame the gesture is performed. If the 'SafeProgress' enumerator object is set to 'FirstLeft' or 'ThirdLeft', as checked within the 'SafeCracking' method every |

| | | | |
|---|---|---|---|
| | | | frame, the Boolean variable ‘safeDialWrongDirection’ is set to ‘true’; this invokes the ‘ExitSafeView’ and ‘SafeReset’ methods within the ‘PuzzleRoomTwo’ script, which resets the SafeProgress’ enumerator object to ‘Active’. Meanwhile, upon a held rightwards or swipe right interaction being performed, the ‘SwipeDIrection’ enumerator object is set to ‘HeldRight’ of ‘Right’. This is used to invoke the ‘SafeDialRotation’ method within the ‘PuzzleRoomTwo’ script; which rotates the safe dial object clockwise for every frame the gesture is performed. If the ‘SafeProgress’ enumerator object is set to ‘Active’ or SecondRight’, as checked within the ‘SafeCracking’ method every frame, the Boolean variable ‘safeDialWrongDirection’ is set to ‘true’; this invokes the ‘ExitSafeView’ and ‘SafeReset’ methods within the ‘PuzzleRoomTwo’ script, which resets the SafeProgress’ enumerator object to ‘Active’. For the rotation and reset procedures of the safe dial object, relevant sounds are invoked by the ‘SafeCracking’ method. |

***Appendix J***:

For outlining the table piece puzzle implementation, delivered in the second puzzle room scene. A step-by-step process demonstrating the puzzles continuity has been compiled, alongside the mechanisms that address each of the processes listed; all of the mechanisms observed in the following table, reside within the ‘InteractionController’ and ‘PuzzleRoomTwo’ classes (see ***Appendix H***).

*Table 2: Puzzle room two, table piece puzzle mechanical implementation, detailing the functional invocation of the puzzle’s components*

| ***Table piece puzzle: mechanical implementation*** | | | |
|---|---|---|---|
| ***Step*** | ***Process*** | ***Mechanisms employed*** | ***How have the mechanisms been addressed?*** |
| 1 | Obtain table piece key, to enable the table piece hanging light to be activated. | Touch-based interaction with the table piece key object, to obtain the table piece key object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the table piece key object is pressed within any screen point occupied by the application. This invokes the ‘GameObjectObtained’ method within the ‘PuzzleRoomTwo’ script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, ‘true’ is returned if the table piece key objects scale is (0, 0, 0), which is further used to identify the ‘TablePieceInteractble’ Boolean variable as ‘true’. |
| 2 | Activate the table piece stand hanging lights, to enable the table piece stand interaction. | Look-based interaction with the table piece stand object, to invoke the lights on animation state of the hanging light object. | Look-based interaction is implemented in the ‘RaycastToObjectsCameraView’ method, within the ‘InteractionController’ class. The method casts a ray from the centre of the first-person cameras viewport and compares the colliding objects tag with the object tag passed in the method. As a method returning a Boolean state, ‘true’ is returned for the table piece stand object seen. This invokes the active animation state for the table piece stand object lights, within the ‘TablePieceSolving’ method, in the ‘PuzzleRoomTwo’ script. |
| 3 | Insert the table piece key, to enable the table globe interaction. | Touch-based interaction with the table piece stand object, to enter the table piece stand interaction view, via the first-person camera object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the table piece stand object is pressed within any screen point occupied by the application. This invokes the ‘EnterTablePieceView’ method within the ‘PuzzleRoomTwo’ script. |

| | | | |
|---|---|---|---|
| | | Touch-based interaction with the table piece key holder object, to insert the table piece key object that was previously obtained, into the table piece key holder object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the table piece key holder object is pressed within any screen point occupied by the application. This is used to alternate the value of the 'TableKeyInserted' Boolean variable, to 'true'; which is then used to invoke the 'ExitTablePieceView' method within the 'PuzzleRoomTwo' script. |
| 4 | Reproduce the table globe animation sequences, to solve the puzzle. | Touch-based interaction with the table globe object, to enter the table globe interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the table globe object is pressed within any screen point occupied by the application. This invokes the 'EnterTableGlobeView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the table globe object, to invoke the first puzzle iteration animation state of the table globe object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the table globe object is pressed within any screen point occupied by the application. This is used to alternate the value of the 'TableGlobePuzzleBegin' Boolean variable, to 'true'; which is then used to invoke the 'IterateTableGlobePuzzle' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the table globe country marker objects, to invoke the table globe country marker material-switch animation and their accompanying audio cues when pressed. Also, to select the county marker objects that formulate the correct sequence, to invoke the next puzzle iteration animation state of the table globe object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if a country marker object is pressed within any screen point occupied by the application. This is used to add string entries to the 'tableGlobeCountryMarkerInput' list of strings; upon a country marker entry being incorrect to the sequence, the 'ResetGlobePuzzle' method is invoked. This resets the table globe puzzle progress variable states. |
| | | Swipe-drag gesture-based interaction when within the table globe interaction view, to invoke the rotary movements of the table globe object. | Swipe-drag gesture-based interaction is governed by the 'InteractionController' class, within the 'RotateAroundAxis' method. The method rotates a passed object in its 'X' and 'Y' axes, relative to its local coordinates. Where the difference accumulated between the gestures current and previous positions, is adapted for being a multiplier to the object's rotation, in the axes specified. |

***Appendix K***:

Proceeding from this passage, the music box puzzle implementation is discussed in relation to its existence within the second puzzle room scene. A step-by-step process that identifies the puzzles structure has been formulated, alongside the mechanisms that address each of the its components; all of the mechanisms referred to in the following table, exist in the 'InteractionController', 'PlayerController' and 'PuzzleRoomTwo' classes (see ***Appendix H***).

*Table 3: Puzzle room two, music box puzzle mechanical implementation, detailing the functional invocation of the puzzle's components*

| ***Music box puzzle: mechanical implementation*** | | | |
|---|---|---|---|
| ***Step*** | ***Process*** | ***Mechanisms employed*** | ***How have the mechanisms been addressed?*** |
| 1 | Activate the music box picture look-interaction, to enable the music box picture touch-interaction. | Look-based interaction with the music box picture object, to invoke the swinging animation state of the music box picture object, the light flickering animation state of the hanging light object and to enter the music box picture interaction view, via the first-person camera object. | Look-based interaction is implemented in the 'RaycastToObjectsCameraView' method, within the 'InteractionController' class. The method casts a ray from the centre of the first-person cameras viewport and compares the colliding objects tag with the object tag passed in the method. As a method returning a Boolean state, 'true' is returned for the music box picture object seen. This invokes the 'EnterMusicBoxPictureVew' and 'MusicBoxPictureAnimate' methods within the 'PuzzleRoomTwo' script. |
| 2 | Activate the carousel music box to enable the music box disc interaction. | Touch-based interaction with the carousel music box object, to enter the carousel music box button interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the carousel music box button object is pressed within any screen point occupied by the application. This invokes the 'EnterCarouselButtonView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the carousel music box button object, to enter the carousel music box cylinder interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the carousel music box cylinder object is pressed within any screen point occupied by the application. This invokes the 'EnterCarouselCylinderView' and 'CarouselMusicBoxAniimate' methods within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the music box disc object, to obtain the music box disc object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the music box disc object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomTwo' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the music box disc objects scale is (0, 0, 0), which is further used to identify the 'MusicBoxDiscObtained' Boolean variable as 'true'. |
| 3 | Obtain the music box, latch key, and wind up key to enable music box stand interaction. | Touch-based interaction with the music box object, to obtain the music box object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the music box object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomTwo' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the music box objects scale is (0, 0, 0), which is further used to identify the 'MusicBoxOtained' Boolean variable as 'true'. |

| | | | |
|---|---|---|---|
| | | Touch-based interaction with the latch key object, to obtain the latch key object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the latch key object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomTwo' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the latch key objects scale is (0, 0, 0), which is further used to identify the 'MusicBoxLatchKeyObatined' Boolean variable as 'true'. |
| | | Touch-based interaction with the wind-up key object, to obtain the wind-up key object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the wind-up key object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomTwo' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the wind-up key objects scale is (0, 0, 0), which is further used to identify the 'MusicBoxWindUpKeyObatined' Boolean variable as 'true'. |
| 4 | Activate the music box stand hanging lights, to enable the music box stand interaction. | Look-based interaction with the music box stand object, to invoke the lights on animation state of the hanging light object. | Look-based interaction is implemented in the 'RaycastToObjectsCameraView' method, within the 'InteractionController' class. The method casts a ray from the centre of the first-person cameras viewport and compares the colliding objects tag with the object tag passed in the method. As a method returning a Boolean state, 'true' is returned for the music box stand object seen. This invokes the active animation state for the music box stand object lights, within the 'MusicBoxSolving' method, in the 'PuzzleRoomTwo' script. |
| 5 | Place the music box, to enable the music box interaction. | Touch-based interaction with the music box stand object, to enter the music box stand interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the music box stand object is pressed within any screen point occupied by the application. This invokes the 'EnterMusicBoxStandView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the music box holder object, to place the music box object that was previously obtained, onto the music box holder object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the music box holder object is pressed within any screen point occupied by the application. This is used to alternate the value of the 'MusicBoxPlaced' Boolean variable, to 'true'; which is then used to invoke the 'ExitMusicBoxStandView' method within the 'PuzzleRoomTwo' script. |
| 6 | Assemble the music box, to solve the puzzle. | Touch-based interaction with the music box collider view objects, to enter the music box item interaction views, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if a music box collider view object is pressed within any screen point occupied by the application. This is used to determine the value of the 'musicBoxPuzzleProgress' enumerator object, which determines the functional invocation for the first-person camera and interaction methods, in the 'MusicBoxAssembling' method, within the 'PuzzleRoomTwo' class. The enumerator object governs this invocation by being the subject of a switch case statement. |

| | | | |
|---|---|---|---|
| | | Swipe-drag gesture-based interaction when within the music box interaction view, to invoke the rotary movements of the camera pivot object. | Swipe-drag gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘RotateAroundAxis’ method. The method rotates a passed object in its ‘X’ and ‘Y’ axes, relative to its local coordinates. Where the difference accumulated between the gestures current and previous positions, is adapted for being a multiplier to the object’s rotation, in the axes specified. |
| | | Pinch gesture-based interactions when within the music box interaction view, to invoke the ‘Z’ dimensional translation of the camera pivot object. | Pinch gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a pinch inwards or outwards interaction being performed, the ‘pinchGesture’ enumerator object is set to ‘PinchInwards’ or ‘PinchOutwards’. This is used to invoke ‘Z’ dimensional translation of the camera pivot object, within the ‘MusicBoxCameraControl’ method, in the ‘PuzzleRoomTwo’ class. |
| | | Swipe-drag gesture-based interaction when within the music box collider view object interaction views, to invoke the rotary movements of select music box items. | Swipe-drag gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘RotateAroundAxis’ method. The method rotates a passed object in its ‘X’ and ‘Y’ axes, relative to its local coordinates. Where the difference accumulated between the gestures current and previous positions, is adapted for being a multiplier to the object’s rotation, in the axes specified. |
| | | Swipe gesture-based interaction when within the music box collider view object interaction views, to invoke the translation of select music box items and accompanying audio cues. | Swipe gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a swipe right or held upwards interaction being performed, the ‘swipeDIrection’ enumerator object is set to ‘Right’ or ‘HeldUp’. This is used to invoke ‘Y’ dimensional translation of the latch key and disc objects, relative to the state of the ‘musicBoxPuzzleProgress’ enumerator object, within the ‘MusicBoxAssembling’ method, in the ‘PuzzleRoomTwo’ class. |
| | | Pinch-rotate gesture-based interaction when within the music box collider view object interaction views, to invoke the rotatory movements of select music box items and accompanying audio cues. | Pinch-rotate gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a pinch-rotate left, pinch-rotate hold left, pinch-rotate right or pinch-rotate hold right interaction being performed, the ‘pinchGesture’ enumerator object is set to ‘RotateLeft’, ‘RotateHeldLeft’, ‘RotateRight’ and ‘RotateHeldRight’. This is used to invoke anti-clockwise and clockwise rotations of the disc needle and wind up key objects, relative to the state of the ‘musicBoxPuzzleProgress’ enumerator object, within the ‘MusicBoxAssembling’ method, in the ‘PuzzleRoomTwo’ class. |
| 7 | Activate the wardrobe doors, to enable the wardrobe hide sequence interactions. | Touch-based interaction with the wardrobe doorknob objects, to invoke the doors open animation state of the wardrobe object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the wardrobe doorknob objects are pressed within any screen point occupied by the application. This is used to alternate the value of the ‘WardrobeDoorsOpenActive’ animator Boolean variable, to ‘true’, within the ‘PuzzleRoomTwo’ script; which enables the player object to enter the wardrobe object. |
| | | Player-based collision interaction with the wardrobe collider object, to invoke the doors close animation state of the wardrobe object. | Collision-based interaction is manged by the ‘PlayerController’ class, within the ‘OnTriggerEnter’ method. The method compares the tag of the object that collides with a player, within a series of if-else statements. Upon the player object colliding with the wardrobe collider object, the ‘forceFirstPersonCameraPerspective’ Boolean variable is set to ‘true’, as well, the ‘puzzleRoomTwoCollision’ string variable is set to the tag of the wardrobe collider object; from within the ‘PlayerController’ class. This prevents the camera perspective from being toggled, and invokes the doors closing animation state of the wardrobe object, and the complementary audio cues, within the ‘MusicBoxHide’ method in the ‘PuzzleRoomTwo’ class. |

| | | | |
|---|---|---|---|
| | | Touch-based interaction with the wardrobe door objects, to invoke the doors open slightly and fully animation states of the wardrobe object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the wardrobe door objects are pressed within any screen point occupied by the application. This is used to alternate the value of the 'WardrobeDoorsSlightlyOpenActive' and 'WardrobeDoorsFullyOpenActive' animator Boolean variables, to 'true', within the 'PuzzleRoomTwo' script; which enables the player object to exit the wardrobe object, upon the wardrobe door objects opening to the full extent. |

***Appendix L***:

In the following table, the mechanical implementation for the piano puzzle is detailed, as advertised in the second puzzle room scene. A step-by-step process for the puzzles development is provided, alongside the mechanisms supporting each of the developments that are discussed; all of the mechanisms that are listed in the following table, reside within the 'InteractionController' and 'PuzzleRoomTwo' classes (see ***Appendix H***).

*Table 4: Puzzle room two, piano  puzzle mechanical implementation, detailing the functional invocation of the puzzle's components*

| ***Piano puzzle: mechanical implementation*** | | | |
|---|---|---|---|
| ***Step*** | ***Process*** | ***Mechanisms employed*** | ***How have the mechanisms been addressed?*** |
| 1 | Activate the bookshelf light to enable the bookshelf book interaction. | Touch-based interaction with the bookshelf light switch object, to invoke the light on animation state of the bookshelf light object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the bookshelf light switch object is pressed within any screen point occupied by the application. This is used to invoke the 'ToggleBookhshelfLight' method, within the 'PuzzleRoomTwo' script; which enables the bookshelf light to be active in the scene. |
| 2 | Obtain the bookshelf book, to enable the bookstand hanging light to be activated. | Touch-based interaction with the bookshelf book object, to enter the bookshelf book interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the bookshelf book object is pressed within any screen point occupied by the application. This invokes the 'EnterPianoMusicSheetBookView' method within the 'PuzzleRoomTwo' script. |
| | | Swipe-drag gesture-based interaction when within the bookshelf book object interaction view, to invoke procedural translation of the bookshelf book object. | Swipe-drag gesture-based interaction for procedural translation is governed by the 'InteractionController' class, within the 'GestureInteraction' method. Upon a swipe down or swipe hold downwards interaction being performed, the 'swipeDirection' enumerator object is set to 'Down' or 'HeldDown'. This is used to invoke 'Z' dimensional translation of the bookshelf book object, within the 'ObtainPianoMusicSheetBook' method, in the 'PuzzleRoomTwo' class. The translation of the bookshelf book object is determined by the gesture interactions difference in current and previous positions; this is calculated in the 'GestureInteraction' method, where the determined difference is assigned to the 'swipeDifference' float variable. |

| | | | |
|---|---|---|---|
| | | Touch-based interaction with the bookshelf book object, to obtain the bookshelf book object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the bookshelf book object is pressed within any screen point occupied by the application. This invokes the ‘GameObjectObtained’ method within the ‘PuzzleRoomTwo’ script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, ‘true’ is returned if the bookshelf book objects scale is (0, 0, 0), which is further used to identify the ‘MusicSheetBookObtained’ Boolean variable as ‘true; which is then used to invoke the ‘ExitPianoMusicSheetBookView’ method within the ‘PuzzleRoomTwo’ script. |
| 3 | Activate the bookstand stand hanging lights, to enable the bookstand stand interaction. | Look-based interaction with the bookstand object, to invoke the lights on animation state of the hanging light object. | Look-based interaction is implemented in the ‘RaycastToObjectsCameraView’ method, within the ‘InteractionController’ class. The method casts a ray from the centre of the first-person cameras viewport and compares the colliding objects tag with the object tag passed in the method. As a method returning a Boolean state, ‘true’ is returned for the bookstand object seen. This invokes the active animation state for the music box stand object lights, within the ‘PianoPuzzleSolving’ method, in the ‘PuzzleRoomTwo’ script. |
| 4 | Place the bookshelf book, to enable the bookshelf book interaction. | Touch-based interaction with the bookstand object, to enter the bookstand interaction view, via the first-person camera object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the bookstand object is pressed within any screen point occupied by the application. This invokes the ‘EnterPianoBookStandView’ method within the ‘PuzzleRoomTwo’ script. |
| | | Touch-based interaction with the bookstand shelf object, to place the bookshelf book object that was previously obtained, onto the bookstand shelf object. | Touch-based interaction is manged by the ‘InteractionController’ class, within the ‘RaycastHitInput’ method. The method compares the tag of the object that a ray is casted to, from within the ‘RaycastToObjectsMouseInteraction’ method. As a method returning a Boolean state, ‘true’ is returned if the bookstand shelf object is pressed within any screen point occupied by the application. This is used to alternate the value of the ‘MusicSheetBookPlaced’ Boolean variable, to ‘true’; which is then used to invoke the ‘MusicSheetBookFlicker’ method within the ‘PuzzleRoomTwo’ script. |
| 5 | Obtain the music sheet, to enable the piano strobe lights to be activated. | Swipe gesture-based interaction when within the bookstand book object interaction view, to invoke the page open and close animation states and accompanying audio cues. | Swipe gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a swipe left or swipe right interaction being performed, the ‘swipeDIrection’ enumerator object is set to ‘Left’ or ‘Right’. This is used to invoke the page open and close animation states for the bookshelf book object, relative to the state of the ‘musicSheetBookCurrentPage’ enumerator object, within the ‘MusicSheetBookFlicker’ method, in the ‘PuzzleRoomTwo’ class. |
| | | Swipe-drag gesture-based interaction when within the bookstand book object interaction view, to invoke procedural translation of the music sheet object. | Swipe-drag gesture-based interaction for procedural translation is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a swipe up or swipe hold upwards interaction being performed, the ‘swipeDirection’ enumerator object is set to ‘Up’ or ‘HeldUp’. This is used to invoke ‘X’ dimensional translation of the music sheet object, within the ‘MusicSheetBookFlicker’ method, in the ‘PuzzleRoomTwo’ class. The translation of the music sheet object is determined by the gesture interactions difference in current and previous positions; this is calculated in the ‘GestureInteraction’ method, where the determined difference is assigned to the ‘swipeDifference’ float variable. |

| | | | |
|---|---|---|---|
| | | Touch-based interaction with the music sheet object, to obtain the music sheet object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the music sheet object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomTwo' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the music sheet objects scale is (0, 0, 0), which is further used to identify the 'MusicSheetObtained' Boolean variable as 'true; which is then used to invoke the 'ExitPianoBookStandView' method within the 'PuzzleRoomTwo' script. |
| 6 | Play the piano to escape the room. | Touch-based interaction with the piano object, to enter the piano interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the piano object is pressed within any screen point occupied by the application. This invokes the 'EnterPianoView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the piano shelf object, to enter the piano shelf interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the piano shelf object is pressed within any screen point occupied by the application. This invokes the 'EnterPianoShelfView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the piano shelf object, to place the music sheet object that was previously obtained, onto the piano shelf object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the piano shelf object is pressed within any screen point occupied by the application. This is used to alternate the value of the 'PianoMusicSheetPlaced' Boolean variable, to 'true'; which is then used to invoke the 'ExitPianoShelfView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the piano key shutter object, to enter the piano key interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the piano key shutter object is pressed within any screen point occupied by the application. This invokes the 'EnterPianoKeyView' method within the 'PuzzleRoomTwo' script. |
| | | Touch-based interaction with the piano key objects, to invoke the piano key played animation states and their accompanying audio cues when pressed. Also, to press the piano key objects that formulate the correct sequence, to complete the puzzle. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if a piano key object is pressed within any screen point occupied by the application. This is used to add string entries to the 'pianoKeyNoteInput' list of strings; upon a piano key object being pressed for the current attempt of the puzzle, if another piano key object is not pressed within five seconds, the 'pianoKeyViewActive' Boolean variable is set to 'false' and the 'pianoKeyNoteReset' Boolean variable is set to 'true'. This is identified by the invocation of the 'CheckPianoNotesPlayed' method, every frame, which utilises the 'pianoKeyNotePlayedTimer' float variable, to accumulate the time for when a piano key object was last interacted with. Alternatively, if the list of strings contains thirteen entries which are determined to be wrong in sequence, as |

| | | | |
|---|---|---|---|
| | | | checked by 'CheckPianoNotesPlayed' every frame, the Boolean variable states are also alternated. This results in the invocation of the 'EnterPianoShelfView' method, which resets the piano puzzle variables states. |
| | | Swipe gesture-based interaction when within the piano key and piano shelf object interaction views, to alternate the active interaction view, via the first-person camera. | Swipe gesture-based interaction is governed by the 'InteractionController' class, within the 'GestureInteraction' method. Upon a swipe hold up or swipe hold down interaction being performed, the 'swipeDIrection' enumerator object is set to 'HeldUp' or 'HeldDown'. This is used to alternate the value of the 'pianoKeyViewActive' Boolean variable within the 'AlternatePianoCamera' method; which is then used to invoke the 'EnterPianoShelfView' method if 'false', and the 'EnterPianoKeyView' if 'true', which are integrated within the 'PianoPuzzleSolving' method in the 'PuzzleRoomTwo' class. |

***Appendix M:***

Below, features the telephone puzzle implementation, as advertised in the first puzzle room scene. A step-by-step process for the puzzles gameplay structure is accounted for, alongside the mechanisms supporting each of the phases that are mentioned; all of the mechanisms that are detailed in the following table, reside within the 'InteractionController', 'SubtitleController' and 'PuzzleRoomOne' classes (see ***Appendix H***).

*Table 5: Puzzle room one, telephone puzzle mechanical implementation, detailing the functional invocation of the puzzle's components*

| ***Telephone puzzle: mechanical implementation*** | | | |
|---|---|---|---|
| ***Step*** | ***Process*** | ***Mechanisms employed*** | ***How have the mechanisms been addressed?*** |
| 1 | Answer the telephone to activate the ornament stand interactions in the scene. | Touch-based interaction with the telephone object, to enter the telephone interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the telephone object is pressed within any screen point occupied by the application. This invokes the 'EnterTelephoneView' method within the 'PuzzleRoomOne' script. |
| | | Swipe gesture-based interaction when within telephone interaction view, to invoke the answer telephone animation state of the telephone earpiece object. | Swipe gesture-based interaction is governed by the 'InteractionController' class, within the 'GestureInteraction' method. Upon a held upwards interaction being performed, the 'swipeDIrection' enumerator object is set to 'HeldUp'. This is used to invoke the 'PickUpTelephone' method within the 'PuzzleRoomOne' script. |
| 2 | Activate the ornament stand hanging lights to enable the face mask and cabinet window interactions. | Look-based interaction with the ornament stand objects, to invoke the lights on animation state of the hanging light objects. | Look-based interaction is implemented in the 'RaycastToObjectsCameraView' method, within the 'InteractionController' class. The method casts a ray from the centre of the first-person cameras viewport and compares the colliding objects tag with the object tag passed in the method. As a method returning a Boolean state, 'true' is returned for each ornament stand object seen. This invokes the active animation state for each of the ornament stand lights, within the 'TelephonePuzzleSolving' method, in the 'PuzzleRoomOne' script. |
| 3 | Break the glass cabinet window to gain access to the cabinet note, for allowing the cabinet note to be obtained. | Touch-based interaction with the glass cabinet window object, to invoke the glass breaking animation of the glass window object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the glass cabinet window object is pressed within any screen point occupied by the |

| | | | |
|---|---|---|---|
| | | | application. This invokes the active animation state of the glass cabinet window object, within the 'PuzzleRoomOne' script. |
| 4 | Obtain the cabinet note to enable clipboard, audio device and Morse code book interactions. | Touch-based interaction with the cabinet note object, to obtain the cabinet note object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the cabinet note object is pressed within any screen point occupied by the application. This invokes the 'GameObjectObtained' method within the 'PuzzleRoomOne' script, which reduces a passed objects scale and sets it to inactive upon reaching (0, 0, 0). As a method returning a Boolean state, 'true' is returned if the cabinet note objects scale is (0, 0, 0), which is further used to identify the 'NoteObtained' Boolean variable as 'true'. |
| 5 | Place the cabinet note onto the clipboard to enable telephone and telephone button interactions. | Touch-based interaction with the clipboard object to enter the clipboard interaction view, via the first-person camera object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the clipboard object is pressed within any screen point occupied by the application. This is used to invoke the 'EnterClipboardView' method within the 'PuzzleRoomOne' script. |
| | | Touch-based interaction with the clipboard object, to place the cabinet note object that was previously obtained, onto the clipboard object. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the clipboard object is pressed within any screen point occupied by the application. This is used to alternate the value of the 'NotePlaced' Boolean variable, to 'true'; which is then used to invoke the 'ExitClipboardView' method within the 'PuzzleRoomOne' script. |
| 6 | Play the audio device audio cue, to auditorily identify the last segment of the telephone number. | Touch-based interaction with the audio device object, to invoke the Morse code audio cue and the Morse code subtitle sequence. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the audio device object is pressed within any screen point occupied by the application. This is used to invoke the Morse code audio cue and accompanying subtitles within the 'TelephonePuzzleSolving' method, in the 'PuzzleRoomOne' script. Subtitle invocation is addressed by the integration of the 'SubtitleSequence' method, which originates from the 'SubtitleController' class. |
| 7 | Read the Morse code translation page of book, to translate the Morse code audio cue emitted by the audio device, to a number format. | Touch-based interaction with the Morse code book object, to enter the Morse code book interaction view, via the first-person camera. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the Morse code book object is pressed within any screen point occupied by the application. This is used to invoke the 'EnterMorseCodeBookView' method within the 'PuzzleRoomOne' script. |
| 8 | Dial the telephone to escape the puzzle room. | Touch-based interaction with the telephone object, to enter the telephone interaction view, via the first-person camera. | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method returning a Boolean state, 'true' is returned if the telephone object is pressed within any screen point occupied by the application. This invokes the 'EnterTelephoneView' method within the 'PuzzleRoomOne' script. |
| | | Touch-based interaction with the telephone button objects, to invoke the telephone button | Touch-based interaction is manged by the 'InteractionController' class, within the 'RaycastHitInput' method. The method compares the tag of the object that a ray is casted to, from within the 'RaycastToObjectsMouseInteraction' method. As a method |

|  |  |  |  |
|---|---|---|---|
|  |  | pressed animation states and their accompanying audio cues when pressed. Also, to dial telephone numbers, to enable the invocation of the answer telephone animation state, for the telephone earpiece object. | returning a Boolean state, ‘true’ is returned if a telephone button object is pressed within any screen point occupied by the application. This is used to add integer entries to the ‘telephoneButtonInput’ list of integers; upon the list accumulating more than eleven entries, the text mesh variable ‘telephoneScreenText’ is reset to ‘“”’, as well, the ‘telephoneButtonInput’ list is emptied. |
|  |  | Swipe gesture-based interaction when within the telephone interaction view, to invoke the answer telephone animation state of the telephone earpiece object and accompanying audio cues. | Swipe gesture-based interaction is governed by the ‘InteractionController’ class, within the ‘GestureInteraction’ method. Upon a held upwards interaction being performed, the ‘swipeDIrection’ enumerator object is set to ‘HeldUp’. This is used to invoke the ‘PickUpTelephone’ method within the ‘PuzzleRoomOne’ script; which is accompanied by one of two audio cues, determined by the value of the ‘TelephoneNumberSequenceCorrect’ Boolean variable. |

## ***Appendix N:***

*Table 6: Unit test cases, user interface elements*

| Case | Summary | Process | Actual result(s) | Expected result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Disclaimer window button appears upon the start button being interacted with. | Interact with the start button when loaded into the loading screen scene. | ‘displayDisclaimer’ returns ‘true’ when the start button is interacted with. | ‘displayDisclaimer’ returns ‘true’ when the start button is interacted with. | 🟩 |
| 2 | Disclaimer window disappears upon the disclaimer window button being interacted with. | Interact with the disclaimer window button when the disclaimer window appears. | ‘displayDisclaimer’ returns ‘false’ when the disclaimer window button is interacted with. | ‘displayDisclaimer’ returns ‘false’ when the disclaimer window button is interacted with. | 🟩 |
| 3 | Pause menu can be navigated upon the pause button being interacted with. | Interact with the pause button when the game session is not paused. | ‘displayPauseMenu’ returns ‘true’ when the pause button is interacted with and returns ‘false’, when the resume button is interacted with. | ‘displayPauseMenu’ returns ‘true’ when the pause button is interacted with and returns ‘false’, when the resume button is interacted with. | 🟩 |
| 4 | Subtitle activeness can be toggled within the pause menu, upon the game session being paused. | Interact with the subtitle toggle button within the pause menu when the game session is paused. | ‘subtitlesActive’ returns ‘false’ when the subtitle button is initially interacted with and returns ‘true’, when interacted with alternatively. | ‘subtitlesActive’ returns ‘false’ when the subtitle button is initially interacted with and returns ‘true’, when interacted with alternatively. | 🟩 |
| 5 | Player flashlight object activeness can be toggled when the flashlight toggle button is being interacted with. | Interact with the flashlight toggle button within the UI canvas when the game session is not paused. | ‘flashlightActive’ returns ‘false’ when the subtitle button is initially interacted with and returns ‘true’, when interacted with alternatively. | ‘flashlightActive’ returns ‘false’ when the subtitle button is initially interacted with and returns ‘true’, when interacted with alternatively. | 🟩 |
| 6 | Camera perspective alternates upon the camera toggle button being interacted with. | Interact with the camera toggle button within the UI canvas when the game session is not paused. | ‘thirdPersonCameraActive’ returns ‘true’ when the camera toggle button is initially interacted with and returns ‘false’, when interacted with alternatively. | ‘thirdPersonCameraActive’ returns ‘true’ when the camera toggle button is initially interacted with and returns ‘false’, when interacted with alternatively. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 7 | Player object traverses upon the joystick touch field being interacted with. | Interact with the joystick touch filed within the UI canvas when the game session is not paused. | 'inputDirection' does not return (0, 0) when the joystick touch field is interacted with. | 'inputDirection' does not return (0, 0) when the joystick touch field is interacted with. | 🟩 |
| 8 | Player object traversal is reset upon a joystick touch field interaction ending. | Interact with the joystick touch filed within the UI canvas when the game session is not paused. | 'inputDirection' returns (0, 0) when the joystick touch field is interacted with and then disassociated with. | 'inputDirection' returns (0, 0) when the joystick touch field is interacted with and then disassociated with. | 🟩 |
| 9 | UI elements become inactive upon the first-person camera entering an object views perspective. | Interact with an object that is interactable and requires first-person perspective change. | 'buttonsEnabled' returns 'false' when the first-person camera enters an object views perspective. | 'buttonsEnabled' returns 'false' when the first-person camera enters an object views perspective. | 🟩 |
| 10 | UI elements become active upon the first-person camera exiting an object views perspective. | Interact with an object that is interactable and requires first-person perspective change. | 'buttonsEnabled' returns 'true' when the first-person camera exits an object views perspective. | 'buttonsEnabled' returns 'true' when the first-person camera exits an object views perspective. | 🟩 |
| 11 | Camera perspective cannot be alternated upon the player object colliding with the wardrobe trigger object, and when the camera toggle button is being interacted with. | Interact with the camera toggle button within the UI canvas when the game session is not paused, and when the player object is colliding with the wardrobe trigger object. | 'thirdPersonCameraActive' returns 'false' when the camera toggle button is interacted with, whilst the player object is colliding with the wardrobe trigger object. | 'thirdPersonCameraActive' returns 'false' when the camera toggle button is interacted with, whilst the player object is colliding with the wardrobe trigger object. | 🟩 |

*Table 7: Unit test cases, player*

| Case | Summary | Process | Actual result(s) | Expected result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Player object cannot jump when not grounded. | Interact with the jump button, whilst the player object is not grounded. | 'playerGrounded' returns 'false' when the player object is mid-air and returns 'true', when the player object is colliding with the grounding plane. | 'playerGrounded' returns 'false' when the player object is mid-air and returns 'true', when the player object is colliding with the grounding plane. | 🟩 |
| 2 | Player jump sound is played when the player object jumps. | Interact with the jump button, whilst the player object is grounded. | 'playJumpSound' returns 'true' when the player object is mid-air and returns 'false', when the player object is colliding with the grounding plane. | 'playJumpSound' returns 'true' when the player object is mid-air and returns 'false', when the player object is colliding with the grounding plane. | 🟩 |
| 3 | Player jump sound is played when the player object is grounded from jumping. | Interact with the jump button, whilst the player object is grounded. | 'playLandSound' returns 'false' when the player object is mid-air and returns 'true', upon the player object colliding with the grounding plane. | 'playLandSound' returns 'false' when the player object is mid-air and returns 'true', upon the player object colliding with the grounding plane. | 🟩 |
| 4 | Player footsteps sound is played in the order of the player objects turning direction, when the payer object does not traverse. | Interact with the joystick touch field, whilst the player object is grounded. | 'playerLeftFootstepSoundPlaying' returns 'false' when the player object is rotating right and returns 'true', upon the player object rotating left when grounded. | 'playerLeftFootstepSoundPlaying' returns 'false' when the player object is rotating right and returns 'true', upon the player object rotating left when grounded. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | Player object can only traverse upon input being enabled. | Interact with objects and enter object view perspectives, attempt to interact with UI canvas and use key input. | 'inputEnabled' returns 'false' when the first-person camera is inside an object view perspective and returns 'true' when the first-person camera is not within an object view perspective. | 'inputEnabled' returns 'false' when the first-person camera is inside an object view perspective and returns 'true' when the first-person camera is not within an object view perspective. | 🟩 |
| 6 | Player idle animation is not invoked upon the game session being paused, and when the first-person camera is within an object view. | Interact with objects and enter object view perspectives, pause the game session via the pause menu UI. | 'playerInsideObjectView' returns 'true' when the first-person camera is inside an object view perspective and returns 'false' when the first-person camera is not within an object view perspective. | 'playerInsideObjectView' returns 'true' when the first-person camera is inside an object view perspective and returns 'false' when the first-person camera is not within an object view perspective. | 🟩 |
| 7 | Player object is identified as being within the wardrobe object, upon colliding with the wardrobe trigger object. | Interact with the jump button and joystick touch field, whilst the player object is grounded and is close by to the wardrobe object. | 'puzzleRoomTwoCollision' identifies as 'WardrobeTrigger' upon the player object colliding with the wardrobe trigger object and identifies as empty when the player object is not colliding with the wardrobe trigger object. | 'puzzleRoomTwoCollision' identifies as 'WardrobeTrigger' upon the player object colliding with the wardrobe trigger object and identifies as empty when the player object is not colliding with the wardrobe trigger object. | 🟩 |

*Table 8: Unit test cases, puzzle*

| Case | Summary | Process | Actual result(s) | Expected result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Game object can be obtained upon being interacted with. Game object becomes inactive upon being obtained. | Interact with an object that is interactable and obtainable. | 'GameObjectObtained()' returns 'true' upon the game objects scale reducing to (0, 0, 0), and returns 'false' whilst the game objects scale is not (0, 0, 0). | 'GameObjectObtained()' returns 'true' upon the game objects scale reducing to (0, 0, 0), and returns 'false' whilst the game objects scale is not (0, 0, 0). | 🟩 |
| 2 | Game objects transform can be identified as being at its prior position and rotation offset. | Compare the position and rotation properties of a game object with position and rotation values. | 'GameObjectAtPreviousTransform()' returns the desired Boolean state passed, and when the game objects position and rotation offset is equal to the position and rotation values passed. | 'GameObjectAtPreviousTransform()' returns the desired Boolean state passed, and when the game objects position and rotation offset is equal to the position and rotation values passed. | 🟩 |
| 3 | Camera objects transform can be identified as being at a position and rotation offset. | Compare the position and rotation properties of the camera with position and rotation values. | 'CameraAtTransform()' returns 'true' upon the camera objects position and rotation being equal to the position and rotation values passed. | 'CameraAtTransform()' returns 'true' upon the camera objects position and rotation being equal to the position and rotation values passed. | 🟩 |
| 4 | Camera objects transform can be stored as a position and rotation offset. | Compare the position and rotation values store before and after the camera objects transform updates. | 'StorePreviousTransform()' returns a vector and quaternion not equal to '0', thereby storing the camera objects position and rotation offset to the position and rotation values passed. | 'StorePreviousTransform()' returns a vector and quaternion not equal to '0', thereby storing the camera objects position and rotation offset to the position and rotation values passed. | 🟩 |
| 5 | Game objects transform can be identified as being at another game | Compare the position and rotation properties of the game object | 'ObjectAtTransform()' returns 'true' upon the game objects transform being identical to the target game objects transform. | 'ObjectAtTransform()' returns 'true' upon the game objects transform being identical to the target game objects transform. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | objects position and rotation. | with the position and rotation values of another game object. | | | 🟩 |

## *Appendix O:*

*Table 9: Black-box test cases, user interface elements*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Disclaimer window appears with the according disclaimer text and disappears when the disclaimer window button is being interacted with, via touch. | Start button is pressed at the initial scene when the game is loaded. The disclaimer window button is pressed when the disclaimer window appears. | Start button is pressed at the initial scene when the game is loaded, the disclaimer GUI window appears in the centre of the screen with the according text, when the disclaimer window (button) is pressed, the disclaimer window disappears and the following scene is loaded. | Start button is pressed at the initial scene when the game is loaded, the disclaimer GUI window appears in the centre of the screen with the according text, when the disclaimer window (button) is pressed, the disclaimer window disappears and the following scene is loaded. | 🟩 |
| 2 | Player object travels upwards and descends gradually when jump button is pressed. | Jump button embedded within the UI is pressed when the player object is grounded. | Player object travels upwards once jump button is pressed; the player object gradually descends due to gravity parameter set. | Player object travels upwards once jump button is pressed; the player object gradually descends due to gravity parameter set. | 🟩 |
| 3 | Player object rotates around its own axis positively in the 'Y' axis when the joystick is held right. | Joystick embedded within the UI is held right via touch and drag gesture. | Player object rotates rightwards around its own axis in the 'Y' axis at a constant rate when joystick is held right. | Player object rotates rightwards around its own axis in the 'Y' axis at a constant rate when joystick is held right. | 🟩 |
| 4 | Player object rotates around its own axis negatively in the 'Y' axis when the joystick is held left. | Joystick embedded within the UI is held left via touch and drag gesture. | Player object rotates leftwards around its own axis in the 'Y' axis at a constant rate when joystick is held left. | Player object rotates leftwards around its own axis in the 'Y' axis at a constant rate when joystick is held left. | 🟩 |
| 5 | Player object traverses' forwards in the current facing direction when joystick is held up. | Joystick embedded within the UI is held up via touch and drag gesture | Player object traverses' forwards in the current facing direction at a constant rate when the when joystick is held upwards. | Player object traverses' forwards in the current facing direction at a constant rate when the when joystick is held upwards. | 🟩 |
| 6 | Player object traverses' backwards in the current facing direction when joystick is held down. | Joystick embedded within the UI is held down via touch and drag gesture. | Player object traverses' backwards in the current facing direction at a constant rate when the when joystick is held downwards. | Player object traverses' backwards in the current facing direction at a constant rate when the when joystick is held downwards. | 🟩 |
| 7 | Camera perspective alternates between the first and third person perspectives when the camera cycle button is pressed. | Camera cycle button embedded within the UI is pressed. | Camera perspective changes from the current perspective to the alternate perspective, upon the camera cycle button being pressed. | Camera perspective changes from the current perspective to the alternate perspective, upon the camera cycle button being pressed. | 🟩 |
| 8 | Player flashlight object toggles active state when flashlight cycle button is pressed. | Flashlight cycle button embedded within the UI is pressed. | Flashlight light source alternates between active and inactive upon the flashlight cycle button being pressed. | Flashlight light source alternates between active and inactive upon the flashlight cycle button being pressed. | 🟩 |
| 9 | Joystick image recentres to its anchored position, upon the interaction between the touch field ending. | Joystick embedded within the UI is interacted with via touch or drag gestures. | Joystick image recentres to the touch field, where it is anchored upon the interaction ending. | Joystick image recentres to the touch field, where it is anchored upon the interaction ending. | 🟩 |
| 10 | Pause menu appears, UI elements become inactive and the | Pause button embedded within the UI is pressed. | Pause menu appears when the pause button is pressed, UI elements are not visible, | Pause menu appears when the pause button is pressed, UI elements are not visible | 🟩 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | game session does not update when pause button is pressed. |  | and the game session does not update. | and the game session does not update. | 🟩 |
| 11 | Pause menu disappears, UI elements become active and the game session updates when the resume button is pressed. | Resume button embedded within the UI is pressed. | Pause menu disappears when the resume button is pressed, UI elements are visible, and the game session updates. | Pause menu disappears when the resume button is pressed, UI elements are visible, and the game session updates. | 🟩 |
| 12 | Subtitle activeness is alternated when subtitle cycle button is pressed. | Subtitles cycle button embedded within the UI is pressed. | Subtitle activeness alternates between active and inactive upon the subtitle cycle button being pressed, subtitle cycle buttons texture updates accordingly. | Subtitle activeness alternates between active and inactive upon the subtitle cycle button being pressed, subtitle cycle buttons texture updates accordingly. | 🟩 |
| 13 | Application closes when application exit button is pressed. | Application exit button embedded within the UI is pressed. | Application closes upon the application exit button being pressed. | Application closes upon the application exit button being pressed. | 🟩 |
| 14 | All UI elements are deactivated upon the player entering an object interaction view. | Player interacts with an interactable rigid object with collider component(s). | All UI element images disappear, and the UI elements buttons become unresponsive, player object movement is not interrupted. | All UI element images become inactive and the UI elements buttons become unresponsive, player object movement is not interrupted. | 🟩 |
| 15 | All UI elements are reactivated upon the player exiting an object interaction view. | Player complete interaction with an interactable rigid object with collider component(s). | All UI element images reappear, and the UI element buttons become responsive again, player object movement can be invoked. | All UI element images reappear, and the UI element buttons become responsive again, player object movement can be invoked. | 🟩 |
| 16 | Upon the player entering the wardrobe rigid object and invoking the hide sequence in the second puzzle room, camera cycle button can be interacted with, but camera perspective does not alternate. | Camera cycle button embedded within the UI is pressed, when player object is situated within the wardrobe rigid object. | Camera cycle button can be pressed and animates within the UI; however, the camera perspective remains within the first-person. | Camera cycle button can be pressed and animates within the UI; however, the camera perspective remains within the first-person. | 🟩 |

*Table 10: Black-box test cases, player*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Player object deflects off objects, does not penetrate through other rigid objects with collider component(s). | Player object traverses towards rigid object with collider component(s) until impact, where it continually collides. | Player object is continually forced from the object it collides with, the player object remains upright and on a surface; player object does not penetrate through object. | Player object is continually forced from the object it collides with, the player object remains upright and on a surface; player object does not penetrate through object. | 🟩 |
| 2 | Player object does not stick to other rigid objects with collider component(s) and physic material(s). | Player object traverses towards rigid object with collider component(s) and physic material(s) until impact, where the player object then jumps and continually traverses towards the object mid-air. | Player object does not stick to other rigid objects with collider component(s) and physic material(s), player gradually falls until being surfaced with the grounding plane. | Player object does not stick to other rigid objects with collider component(s) and physic material(s), player gradually falls until being surfaced with the grounding plane. | 🟩 |
| 3 | Player jumping and landing sounds as well as animation(s) are invoked upon | Jump button embedded within the UI is pressed when the player object is grounded. | Player object traverses upwards and the jumping animation is invoked as well as the jumping sound, upon the player traversing | Player object traverses upwards and the jumping animation is invoked as well as the jumping sound, upon the player traversing | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | the player object jumping. | | downwards and becoming grounded, the landing sound is played. | downwards and becoming grounded, the landing sound is played. | 🟩 |
| 4 | Player objects movement speed and breathing sounds volume and pitch properties adjust with the offset of the joystick. | Joystick embedded within the UI is held down via touch and drag gesture. | Player objects movement speed and breathing sound sounds volume and pitch properties increase when the joystick is held further away from its anchored position, creating the impression of a faster movement pace. | Player objects movement speed and breathing sound sounds volume and pitch properties increase when the joystick is held further away from its anchored position, creating the impression of a faster movement pace. | 🟩 |
| 5 | Player objects animation controller enables the player to jump, turn and traverse in the facing direction simultaneously. | Joystick embedded within the UI is held down via touch and drag gesture, whilst the jump button embedded within the UI is pressed when the player object is grounded simultaneously. | Player objects animation controller utilises blend tree animation states to jump, turn and traverse in the facing direction simultaneously, where all sounds accompanying the player objects movement are invoked also. | Player objects animation controller utilises blend tree animation states to jump, turn and traverse in the facing direction simultaneously, where all sounds accompanying the player objects movement are invoked also. | 🟩 |
| 6 | Player object enters idle animation state when there is no interaction with the UI elements. | Launch the application and load into a scene. Also, interact with the joystick or jump button. | Player objects animation controller invokes the idle state, where the corresponding breathing sound is played, previous animation state sounds are transitioned from. | Player objects animation controller invokes the idle state, where the corresponding breathing sound is played, previous animation state sounds are transitioned from. | 🟩 |
| 7 | Player object turns and traverses and invokes footstep sounds to simulate walking, only when grounded. | Joystick embedded within the UI is held down via touch and drag gesture, whilst the player object is grounded. | Player footstep sounds are invoked when the player object is grounded and is traversing or turning. Meanwhile, footstep sounds are not invoked whilst player object is not grounded, not turning, or not traversing. | Player footstep sounds are invoked when the player object is grounded and is traversing or turning. Meanwhile, footstep sounds are not invoked whilst player object is not grounded, not turning, or not traversing. | 🟩 |
| 8 | Player object is facing a rigid object relatively close with collider component(s), player animation sounds are not invoked but are otherwise. | Player object traverses towards rigid object with collider component(s) until they are situated close together. | Player animation sounds are not invoked upon the player object becoming within close facing range of a rigid object with collider component(s). However, player animations sounds are invoked upon the player object not being within close facing range of said object(s). | Player animation sounds are not invoked upon the player object becoming within close facing range of a rigid object with collider component(s). However, player animations sounds are invoked upon the player object not being within close facing range of said object(s). | 🟩 |
| 9 | Player object enables door rigid object animations to be triggered upon colliding with door trigger component(s), when a puzzle room is complete or when transitioning to the next room in the sequence. | Player object traverses towards door rigid object with trigger component(s) until impact, the door animations will be or not be invoked, depending on the puzzle room completion states. | Player object is able to traverse beyond the door rigid object upon a puzzle rooms interactions being complete or when transitioning to the next puzzle room in the sequence. Meanwhile, player object is not able to traverse beyond the door rigid object when a puzzle rooms interactions are not complete, or when attempting to transition to puzzle rooms that are not consecutive to the room completed prior. | Player object is able to traverse beyond the door rigid object upon a puzzle rooms interactions being complete or when transitioning to the next puzzle room in the sequence. Meanwhile, player object is not able to traverse beyond the door rigid object when a puzzle rooms interactions are not complete, or when attempting to transition to puzzle rooms that are not consecutive to the room completed prior. | 🟩 |
| 10 | Player object is only able to transition to the jumping state upon being grounded. | Jump button embedded within the UI is pressed continually when the player is grounded and mid-air. | Player object is able to jump from the initial state of being grounded and when being grounded from landing to a jump prior. Player object is not able to jump repetitively otherwise. | Player object is able to jump from the initial state of being grounded and when being grounded from landing to a jump prior. Player object is not able to jump repetitively otherwise. | 🟩 |
| 11 | Player object is only able to enter the | Jump button embedded within the UI is pressed | Player object is able to jump into wardrobe but unable to | Player object is able to jump into wardrobe but unable to | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | wardrobe rigid object in the second puzzle room upon by jumping. | whilst the joystick embedded within the UI is held in the direction of the wardrobe rigid object. | without jumping. Player object will continually collide with wardrobe shelf rigid object when joystick is held towards the direction of the wardrobe, when the player object is not jumping. | without jumping. Player object will continually collide with wardrobe shelf rigid object when joystick is held towards the direction of the wardrobe, when the player object is not jumping. | 🟩 |
| 12 | Upon the player entering the wardrobe rigid object and invoking the hide sequence in the second puzzle room, wardrobe door animation is invoked upon colliding with wardrobe trigger component. | Jump button embedded within the UI is pressed whilst the joystick embedded within the UI is held in the direction of the wardrobe rigid object, player objects enters wardrobe rigid object. | Player object enters the wardrobe rigid object and upon impact with the wardrobe trigger component. Wardrobe door animation is invoked where the player object is encapsulated within the wardrobe rigid object. | Player object enters the wardrobe rigid object and upon impact with the wardrobe trigger component. Wardrobe door animation is invoked where the player object is encapsulated within the wardrobe rigid object. | 🟩 |

*Table 11: Black-box test cases, object interaction within puzzle room two, safe puzzle*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Picture rigid object covering safe rigid object animates, upon interacting with the picture rigid object via touch and whilst being faced towards. | Touch the picture rigid object when the first-person camera aligns the picture rigid object in the centre of the FOV. | Picture rigid object covering safe rigid object animates, when first-person camera aligns the picture rigid object in the centre of the FOV and is touched upon. | Picture rigid object covering safe rigid object animates, when first-person camera aligns the picture rigid object in the centre of the FOV and is touched upon. | 🟩 |
| 2 | First-person camera enters safe object view, upon the safe rigid object's being interacted with via touch and are being faced towards. | Touch the safe rigid objects when the first-person camera aligns the safe rigid object in the centre of the FOV and when the safe rigid object is not centred. | First-person camera moves and rotates towards the safe view object that simulates the perspective for the safe object interaction, upon the safe objects being touched and whilst the first-person camera is facing the safe objects. Otherwise the first-person camera remains at the position and rotation of the player object. | First-person camera moves and rotates towards the safe view object that simulates the perspective for the safe object interaction, upon the safe objects being touched and whilst the first-person camera is facing the safe objects. Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 3 | Safe dial rigid object rotates clockwise upon the player performing a swipe left or held left gesture interaction, when within the safe view objects perspective. | Swipe left or hold left on the devices screen when within the safe view objects perspective. | Safe dial rigid object rotates clockwise for one frame when a swipe left gesture interaction is performed within the safe view object perspective. The safe dial rigid objects rotation is continued for every frame that the gesture is held left. | Safe dial rigid object rotates clockwise for one frame when a swipe left gesture interaction is performed within the safe view object perspective. The safe dial rigid objects rotation is continued for every frame that the gesture is held left. | 🟩 |
| 4 | Safe dial rigid object rotates anti-clockwise upon the player performing a swipe right or held right gesture interaction, when within the safe view objects perspective. | Swipe right or hold right on the devices screen when within the safe view objects perspective. | Safe dial rigid object rotates anti-clockwise for one frame when a swipe right gesture interaction is performed within the safe view object perspective. The safe dial rigid objects rotation is continued for every frame that the gesture is held right. | Safe dial rigid object rotates anti-clockwise for one frame when a swipe right gesture interaction is performed within the safe view object perspective. The safe dial rigid objects rotation is continued for every frame that the gesture is held right. | 🟩 |
| 5 | Safe dial rigid objects rotation resets upon a swipe left or held left being performed, when the safe dial rigid object is expected to | Swipe left or hold left on the devices screen when within the safe view objects perspective and when the current safe puzzle iteration expects an anti-clockwise interaction. | Safe dial rigid objects rotation rotates towards a neutral degree, the starting rotation of the safe dial rigid object. | Safe dial rigid objects rotation rotates towards a neutral degree, the starting rotation of the safe dial rigid object. | 🟩 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | be rotating anti-clockwise. |  |  |  | 🟩 |
| 6 | Safe dial rigid objects rotation resets upon a swipe right or held right being performed, when the safe dial rigid object is expected to be rotating clockwise. | Swipe right or hold right on the devices screen when within the safe view objects perspective and when the current safe puzzle iteration expects a clockwise interaction. | Safe dial rigid objects rotation rotates towards a neutral degree, the starting rotation of the safe dial rigid object. | Safe dial rigid objects rotation rotates towards a neutral degree, the starting rotation of the safe dial rigid object. | 🟩 |
| 7 | First-person camera exits safe object view, upon the safe dial rigid objects rotation being reset. | Swipe left or hold left on the devices screen when within the safe view objects perspective and when the current safe puzzle iteration expects an anti-clockwise interaction, or alternative. | First-person camera moves and rotates back towards the player object whilst the safe dial rigid objects rotation rotates towards a neutral degree. | First-person camera moves and rotates back towards the player object whilst the safe dial rigid objects rotation rotates towards a neutral degree. | 🟩 |
| 8 | Safe dial puzzle progress is reset parallel to the safe dial rigid objects rotation being reset, a swipe left or held left interaction is expected from the first safe puzzle iteration. | Swipe left or hold left on the devices screen when within the safe view objects perspective, upon entering the safe objects view after the safe dial rigid object has reset. | Swipe left or held left interaction is required to rotate the safe dial clockwise, as the first safe puzzle iteration. Upon a swipe right or held right interaction the safe dial rigid objects rotation resets. | Swipe left or held left interaction is required to rotate the safe dial clockwise, as the first safe puzzle iteration. Upon a swipe right or held right interaction the safe dial rigid objects rotation resets. | 🟩 |
| 9 | First-person camera only enters safe object view, upon the safe rigid object's being interacted with via touch and are being faced towards, as well as when the safe dial rigid objects rotation has been reset. | Touch the safe rigid object when the first-person camera aligns the safe rigid object in the centre of the FOV and when the safe rigid object is not centred, whilst when the safe dial rigid object is rotating and stationary. | First-person camera moves and rotates towards the safe view object that simulates the perspective for the safe object interaction, only when the safe dial rigid objects rotation has been reset and is stationary, and the safe objects are being faced towards and have been touched. | First-person camera moves and rotates towards the safe view object that simulates the perspective for the safe object interaction, only when the safe dial rigid objects rotation has been reset and is stationary, and the safe objects are being faced towards and have been touched. | 🟩 |
| 10 | First-person camera exits safe object view, upon the safe puzzle being complete via swipe right or held right gesture interaction. | Swipe right or hold right on the devices screen when within the safe view objects perspective, until the safe dial rigid objects rotation equates to the correct rotation for the last iteration of the puzzle. | First-person camera moves and rotates back towards the player object whilst the safe rigid object animates. | First-person camera moves and rotates back towards the player object whilst the safe rigid object animates. | 🟩 |
| 11 | First-person camera does not enter safe object view, upon the safe rigid objects being interacted with via touch and are being faced towards when the safe puzzle is complete. | Touch the safe rigid object when the first-person camera aligns the safe rigid object in the centre of the FOV and when the safe rigid object is not centred, upon completing the safe puzzle. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the safe view objects perspective. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the safe view objects perspective. | 🟩 |

*Table 12: Black-box test cases, object interaction within puzzle room two, table puzzle*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Table piece key object can be obtained when being interacted with via touch, upon | Player object traverses towards safe rigid object until relatively close together and touches the table piece key object. | Table piece key objects scale reduces to being invisible upon being touched, where the item obtained sound is | Table piece key objects scale reduces to being invisible upon being touched, where the item obtained sound is | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | the safe puzzle being complete, the table piece key object animates as disappearing and emits sound | | then played to signal its possession. | then played to signal its possession. | 🟩 |
| 2 | Table piece hanging light object animates and emits light flickering sounds upon the table piece key object being obtained and when the table piece stand rigid objects are being faced towards. | Player object traverses towards table piece stand rigid objects until relatively close together and facing the table piece stand rigid object. | Table piece hanging light object animates and emits light flickering sounds, when the player is facing the table piece stand rigid objects within close range, and the table piece key object has been obtained. | Table piece hanging light object animates and emits light flickering sounds, when the player is facing the table piece stand rigid objects within close range, and the table piece key object has been obtained. | 🟩 |
| 3 | First-person camera enters table piece stand object view, upon the table globe rigid object being interacted with via touch and are being faced towards. | Touch the table globe rigid object when the first-person camera aligns the table globe object in the centre of the FOV and when the table globe rigid object is not centred. | First-person camera moves and rotates towards the table piece stand view object that simulates the perspective for the table piece stand object interaction, upon the table globe object being touched and whilst the first-person camera is facing the table globe object. Otherwise the first-person camera remains at the position and rotation of the player object. | First-person camera moves and rotates towards the table piece stand view object that simulates the perspective for the table piece stand object interaction, upon the table globe object being touched and whilst the first-person camera is facing the table globe object. Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 4 | Table piece key object can be placed upon the first-person camera entering the table piece stand object view, where the table globe rigid object and table piece light plane objects active animation states are invoked. | Touch the table piece key holder object when within the table piece stand view objects perspective. | Table piece key is placed and appears in the scene within the table piece holder object. Table globe rigid object and table piece light plane objects active animation states are invoked. | Table piece key is placed and appears in the scene within the table piece holder object. Table globe rigid object and table piece light plane objects active animation states are invoked. | 🟩 |
| 5 | First-person camera exits table piece stand object view, upon the table piece key object being placed into the table piece stand holder object. | Touch the table piece stand holder object when within the table piece stand view objects perspective. | First-person camera exits the table piece stand object view upon the table piece key object being placed into the table piece stand holder object. | First-person camera exits the table piece stand object view upon the table piece key object being placed into the table piece stand holder object. | 🟩 |
| 6 | First-person camera enters table globe object view, upon the table globe rigid object being interacted with via touch and are being faced towards, after the table piece key object being placed. | Touch the table globe rigid object when the first-person camera aligns the table globe object in the centre of the FOV and when the table globe rigid object is not centred, after the table piece key object has been placed. | First-person camera moves and rotates towards the table globe view object that simulates the perspective for the table globe object interaction. | First-person camera moves and rotates towards the table globe view object that simulates the perspective for the table globe object interaction. | 🟩 |
| 7 | Table globe pivot object rotates to the first-person camera's rotation inversed in the 'Y' axis, upon the first-person camera entering the table globe object view. | Touch the table globe rigid object when the first-person camera aligns the table globe object in the centre of the FOV and when the table globe rigid object is not centred, after the table piece key object has been placed. | Table globe pivot object rotates to the first-person camera's rotation inversed in the 'Y' axis, until it aligns, whilst the table globe rigid object is animating. | Table globe pivot object rotates to the first-person camera's rotation inversed in the 'Y' axis, until it aligns, whilst the table globe rigid object is animating. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 8 | Table globe rigid object is rotatable upon the table globe rigid object transitioning from the first puzzle iteration animation to the static animation state, and when being interacted with via touch and drag gestures. | Touch and drag on the devices screen when the table globe rigid object is animating the first puzzle iteration sequence, and when the table globe rigid object enters the static animation state. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | 🟩 |
| 9 | Table globe rigid object is rotatable upon the table globe rigid object transitioning from the second puzzle iteration animation to the static animation state. | Touch and drag on the devices screen when the table globe rigid object is animating the second puzzle iteration sequence, and when the table globe rigid object enters the static animation state. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | 🟩 |
| 10 | Table globe rigid object is rotatable upon the table globe rigid object transitioning from the third puzzle iteration animation to the static animation state. | Touch and drag on the devices screen when the table globe rigid object is animating the third puzzle iteration sequence, and when the table globe rigid object enters the static animation state. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | Table globe rigid object rotates via touch and drag interaction when the table globe rigid object is within the static animation state, otherwise the table globe rigid object remains stationary. | 🟩 |
| 11 | Table globe rigid object animates first table puzzle iteration upon being interacted with via touch and being within the static animation state. | Touch the table globe rigid object when within the table globe view object's perspective, whilst when the table globe rigid object is rotating and stationary. | Table globe rigid object animates first table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | Table globe rigid object animates first table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | 🟩 |
| 12 | Table globe rigid object animates second table puzzle iteration upon being interacted with via touch and being within the static animation state. | Touch the table globe rigid object when within the table globe view object's perspective, whilst when the table globe rigid object is rotating and stationary. | Table globe rigid object animates second table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | Table globe rigid object animates second table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | 🟩 |
| 13 | Table globe rigid object animates third table puzzle iteration upon being interacted with via touch and being within the static animation state. | Touch the table globe rigid object when within the table globe view object's perspective, whilst when the table globe rigid object is rotating and stationary. | Table globe rigid object animates third table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | Table globe rigid object animates third table puzzle iteration, when the table globe rigid object is stationary and is interacted with via touch. | 🟩 |
| 14 | Table globe rigid objects rotation is reset to its static animation state, upon the incorrect sequence of country markers being selected. | Touch the globe country marker rigid objects when the table piece globe rigid object is rotatable. | Table globe rigid objects rotation is reset towards a neutral degree. Table globe transitions to the static animation state, where the globe is no longer rotatable. | Table globe rigid objects rotation is reset towards a neutral degree. Table globe transitions to the static animation state, where the globe is no longer rotatable. | 🟩 |
| 15 | Table puzzle progress is reset parallel to the table globe rigid objects rotation being reset, the first table puzzle iteration animation is invoked upon the | Touch the globe country marker rigid objects when the table piece globe rigid object is rotatable. | Touching the table globe rigid object is required to start the first table puzzle iteration, where the first table puzzle iteration animation is invoked. Upon touching a country marker incorrectly when the table | Touching the table globe rigid object is required to start the first table puzzle iteration, where the first table puzzle iteration animation is invoked. Upon touching a country marker incorrectly when the table | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | table globe rigid object transitioning to the static animation state and being touched upon. | | globe rigid object is rotatable, the table globe rigid objects rotation resets. | globe rigid object is rotatable, the table globe rigid objects rotation resets. | 🟩 |
| 16 | Table globe country marker rigid objects animate and emit sounds when interacted with via touch, when the table piece globe rigid object is rotatable. | Touch the globe country marker rigid objects when the table piece globe rigid object is rotatable and not rotatable. | Table globe country marker rigid objects animate and emit sounds when the table piece globe rigid object is not rotating and are interacted with via touch. | Table globe country marker rigid objects animate and emit sounds when the table piece globe rigid object is not rotating and are interacted with via touch. | 🟩 |
| 17 | Table globe country marker rigid objects cannot be selected multiple times for the current attempt of the puzzle iteration. | Touch the globe country marker rigid objects multiple times when the table globe rigid object is rotatable. | Table globe country marker rigid objects remain within the same animation state, maintain the same material properties, and do not emit sounds. The table globe rigid object does not reset to the initial iteration of the table puzzle. | Table globe country marker rigid objects remain within the same animation state, maintain the same material properties, and do not emit sounds. The table globe rigid object does not reset to the initial iteration of the table puzzle. | 🟩 |
| 18 | Table globe rigid object transitions to the complete animation state and emits audio, table piece light plane objects inactive animation states are invoked, as well, the table puzzle complete sound is played. | Touch the table globe country marker rigid objects in the correct sequence when the table globe rigid object is rotatable, reach the third table puzzle iteration. | Table globe rigid object transitions to the complete animation state, upon the last correct country marker in the third table puzzle iteration being touched. Table globe rigid object emits sound signalling the completion of the table puzzle. | Table globe rigid object transitions to the complete animation state, upon the last correct country marker in the third table puzzle iteration being touched. Table globe rigid object emits sound signalling the completion of the table puzzle. | 🟩 |
| 19 | First-person camera exits table globe object view, upon the table puzzle being complete. | Touch the table globe country marker rigid objects in the correct sequence when the table globe rigid object is rotatable, reach the third table puzzle iteration. | First-person camera exits the table globe object view, upon the final globe country marker rigid object being touched and table globe rigid object finishes animating the complete state. | First-person camera exits the table globe object view, upon the final globe country marker rigid object being touched and table globe rigid object finishes animating the complete state. | 🟩 |
| 20 | First-person camera does not enter table globe or table piece object views, upon the table globe rigid object being interacted with via touch and are being faced towards when the table puzzle is complete. | Touch the table globe and table piece rigid objects when the first-person camera aligns the table globe or table piece rigid objects in the centre of the FOV and when the table globe or table piece rigid objects are not centred, upon completing the table puzzle. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the table globe or table piece view object perspectives. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the table globe or table piece view object perspectives. | 🟩 |

*Table 13: Black-box test cases, object interaction within puzzle room two, music box puzzle*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | First-person camera enters the music box picture rigid object view, upon the table puzzle complete sound finish playing  and being faced towards. | Player object traverses towards music box picture rigid object until relatively close together whilst facing the music box picture rigid object, when the table puzzle complete sound is playing and finishes playing also. | First-person camera moves and rotates towards the music box picture view object that simulates the perspective for the table piece object interaction, upon the table puzzle complete sound finish playing and whilst the first-person camera is facing the music box picture object. | First-person camera moves and rotates towards the music box picture view object that simulates the perspective for the table piece object interaction, upon the table puzzle complete sound finish playing and whilst the first-person camera is facing the music box picture object. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | | | Otherwise the first-person camera remains at the position and rotation of the player object. | Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 2 | Music box picture rigid object animates upon the first-person camera entering the music box picture object view. | Player object traverses towards music box picture rigid object until relatively close together whilst facing the music box picture rigid object, when the table puzzle complete sound has finished playing. | Music box picture object transitions to the active animation state, where the music box picture object animates a narrative scene and emits audio. | Music box picture object transitions to the active animation state, where the music box picture object animates a narrative scene and emits audio. | 🟩 |
| 3 | Music box picture light object animates upon the first-person camera entering the music box picture object view. | Player object traverses towards music box picture rigid object until relatively close together whilst facing the music box picture rigid object, when the table puzzle complete sound has finished playing. | Music box picture light object transitions to the active animation state, where the music box picture light object animates parallel to the music box picture object. | Music box picture light object transitions to the active animation state, where the music box picture light object animates parallel to the music box picture object. | 🟩 |
| 4 | First-person camera exits the music box picture rigid object view, upon the music box picture rigid object transitioning to the static animation state. | Player object traverses towards music box picture rigid object until relatively close together whilst facing the music box picture rigid object, when the table puzzle complete sound has finished playing. | First-person camera exits the music box picture object view, upon the music box picture object transitioning to the static animation state. | First-person camera exits the music box picture object view, upon the music box picture object transitioning to the static animation state. | 🟩 |
| 5 | Music box picture rigid object animates towards the grounding plane upon touch interaction. | Player object traverses towards music box picture rigid object until relatively close together, music box picture object is touched. | Music box picture object transitions to the active animation state, where the music box picture object animates towards the grounding plane upon touching the music box picture object. | Music box picture object transitions to the active animation state, where the music box picture object animates towards the grounding plane upon touching the music box picture object. | 🟩 |
| 6 | Carousel music box rigid object animates upon the music box picture rigid object transitioning to the static animation state. | Player object traverses towards carousel music box rigid object until the carousel music box rigid object is within the active camera’s frustum. | Carousel music box object transitions to the active animation state, where the carousel music box object shakes and emits audio to attract player attention. | Carousel music box object transitions to the active animation state, where the carousel music box object shakes and emits audio to attract player attention. | 🟩 |
| 7 | Music box picture light object animates upon the music box picture rigid object animating towards the grounding plane. | Player object traverses towards music box picture rigid object until relatively close together, music box picture object is touched. | Music box picture light object transitions to the active animation state, where the music box picture light object animates parallel to the music box picture object. | Music box picture light object transitions to the active animation state, where the music box picture light object animates parallel to the music box picture object. | 🟩 |
| 8 | Music box object can be obtained when being interacted with via touch, upon the music box picture rigid object transitioning to the static animation state, the music box object animates as disappearing and emits sound. | Player object traverses towards music box object until relatively close together and touches the music box object. | Music box objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Music box objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 9 | Music box wind up key object can be | Player object traverses towards music box wind | Music box wind up key objects scale reduces to | Music box wind up key objects scale reduces to | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | obtained when being interacted with via touch, upon the music box picture rigid object transitioning to the static animation state, the music box wind up key object .animates as disappearing and emits sound. | up key object until relatively close together and touches the music box wind up object. | being invisible upon being touched, where the item obtained sound is then played to signal its possession. | being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 10 | Music box latch key object can be obtained when being interacted with via touch, upon the music box picture rigid object transitioning to the static animation state, the music box latch key object animates as disappearing and emits sound. | Player object traverses towards music box latch key object until relatively close together and touches the music box latch key object. | Music box latch key objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Music box latch key objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 11 | First-person camera enters the carousel music box button rigid object view, upon the music box picture rigid object transitioning to the static animation state and the carousel music box rigid object being interacted with, via touch. | Player object traverses towards carousel music box rigid objects until relatively close together and touches the carousel music box rigid objects. | First-person camera moves and rotates towards the carousel music box button view object that simulates the perspective for the carousel music box button object interaction, upon touching the carousel music box objects and the music box picture object transitioning to the static animation state. Otherwise the first-person camera remains at the position and rotation of the player object. | First-person camera moves and rotates towards the carousel music box button view object that simulates the perspective for the carousel music box button object interaction, upon touching the carousel music box objects and the music box picture object transitioning to the static animation state. Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 12 | First-person camera enters the carousel music box cylinder rigid object view, upon the carousel music box button rigid object being interacted with, via touch. | Touch the carousel music box button rigid object when within the carousel music box button view object's perspective. | First-person camera moves and rotates towards the carousel music box cylinder view object that simulates the perspective for the carousel music box cylinder object, upon touching the carousel music box button object. Otherwise the first-person camera remains at the position and rotation of the player object. | First-person camera moves and rotates towards the carousel music box cylinder view object that simulates the perspective for the carousel music box cylinder object, upon touching the carousel music box button object. Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 13 | Carousel music box rigid object transitions to the active animation state where its materials properties are updated and emits sound, upon the carousel music box button rigid object being interacted with, via touch. | Touch the carousel music box button rigid object when within the carousel music box button view object's perspective. | Carousel music box object transitions to the active animation state where its material properties update and emits sound, upon the carousel music box button object being touched. | Carousel music box object transitions to the active animation state where its material properties update and emits sound, upon the carousel music box button object being touched. | 🟩 |
| 14 | First-person camera exits the carousel music box cylinder rigid object view, | Touch the carousel music box button rigid object when within the carousel | First-person camera exits the carousel music box object view, upon the carousel music box object | First-person camera exits the carousel music box object view, upon the carousel music box object | Inconsistent working order (not explainable) |

| | | | | | |
|---|---|---|---|---|---|
| | upon the carousel music box rigid object transitioning to the static animation state. | music box button view object’s perspective. | transitioning to the static animation state. | transitioning to the static animation state. | 🟨 |
| 15 | Music box disc object can be obtained when being interacted with via touch, upon the carousel music box rigid object transitioning to the static animation state, the music box disc object animates as disappearing and emits sound. | Player object traverses towards music box disc object until relatively close together and touches the music box disc object. | Music box disc objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Music box disc objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 16 | First-person camera does not enter carousel music box object view, upon the carousel music box button rigid object being interacted with via touch when the carousel music box rigid object interaction is complete. | Touch the carousel music box rigid object upon completing the carousel music box interaction. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the carousel music box view objects perspective. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the carousel music box view objects perspective. | 🟩 |
| 17 | Music box stand hanging light object animates and emits light flickering sounds upon the music box item objects being obtained and when the music box stand rigid object is being faced towards. | Player object traverses towards music box stand rigid object until relatively close together and facing the music box stand rigid object. | Music box stand hanging light object animates and emits light flickering sounds, when the player is facing the music box stand rigid object within close range, and the music box item objects have been obtained. | Music box stand hanging light object animates and emits light flickering sounds, when the player is facing the music box stand rigid object within close range, and the music box item objects have been obtained. | 🟩 |
| 18 | First-person camera enters the music box stand object view, upon the music box stand rigid object being interacted with via touch and being faced towards. | Player object traverses towards music box stand rigid object until relatively close together whilst facing the music box stand rigid object and touching the music box stand object. | First-person camera moves and rotates towards the music box stand view object that simulates the perspective for the music box stand object, upon touching it. Otherwise the first-person camera remains at the position and rotation of the player object. | First-person camera moves and rotates towards the music box stand view object that simulates the perspective for the music box stand object, upon touching it. Otherwise the first-person camera remains at the position and rotation of the player object. | 🟩 |
| 19 | Music box object can be placed upon the first-person camera entering the music box stand object view, where the music box stand objects active animation state is invoked. | Touch the music box stand holder object when within the music box stand view objects perspective. | Music box object is placed and appears in the scene within the music box stand holder object, upon the music box holder object being touched. Music box stand objects active animation states is invoked. | Music box object is placed and appears in the scene within the music box stand holder object, upon the music box holder object being touched. Music box stand objects active animation states is invoked. | 🟩 |
| 20 | First-person camera exits music box stand object view, upon the music box object being placed into the music box stand holder object. | Touch the music box stand holder object when within the music box stand view objects perspective. | First-person camera exits the music box stand object view upon the music box object being placed into the music box holder object. | First-person camera exits the music box stand object view upon the music box object being placed into the music box holder object. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 21 | First-person camera enters music box rigid object view, upon the music box rigid object being interacted with via touch, after the music box object being placed. | Touch the music box rigid object after the music box object has been placed and when the music box object is being placed. | First-person camera moves and rotates towards the music box view object that simulates the perspective for the music box object interaction when the music box object has been placed. | First-person camera moves and rotates towards the music box view object that simulates the perspective for the music box object interaction when the music box object has been placed. | 🟩 |
| 22 | First-person camera can be rotated around music box pivot object view, upon the first-person camera entering the music box object view, and when being interacted with via touch and drag gesture interactions. | Touch and drag on the devices screen when the first-person camera enters the music box view objects perspective and when the first-person camera is entering the music box view objects perspective. | First-person camera rotates via touch and drag interaction when the first-person camera is within the music box view objects perspective, otherwise the first-person camera remains stationary. | First-person camera rotates via touch and drag interaction when the first-person camera is within the music box view objects perspective, otherwise the first-person camera remains stationary. | 🟩 |
| 23 | First-person camera enters music box latch key collider object view, upon the music box latch key collider view object being interacted with via touch, when the music box latch keyhole object is visible, and the music box puzzle state is neutral. | Touch the music box latch key collider view object after the music box view objects perspective has been entered and when the music box view objects perspective is being entered. | First-person camera moves and rotates towards the music box latch key collider view object that simulates the perspective for the music box latch key object interaction. | First-person camera moves and rotates towards the music box latch key collider view object that simulates the perspective for the music box latch key object interaction. | 🟩 |
| 24 | First-person camera enters initial music box wind up key collider object view, upon the music box wind up key collider view object being interacted with via touch, when the music box wind up keyhole object is visible, and the music box puzzle state is neutral. | Touch the music box wind up key collider view object. | First-person camera moves and rotates towards the music box wind up key collider view object that simulates the perspective for the music box latch key object interaction. | First-person camera moves and rotates towards the music box wind up key collider view object that simulates the perspective for the music box latch key object interaction. | 🟩 |
| 25 | First-person camera enters music box lid collider object view, upon the music box lid collider view object being interacted with via touch, when the music box lid object is visible, and once the music box latch key object interaction is complete. | Touch the music box lid collider view object before and after the music box latch key object interactions are complete. | First-person camera moves and rotates towards the music box lid collider view object that simulates the perspective for the music box lid object interaction when the music box latch key object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | First-person camera moves and rotates towards the music box lid collider view object that simulates the perspective for the music box lid object interaction when the music box latch key object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | 🟩 |
| 26 | First-person camera enters music box disc spindle collider object view, upon the music box disc spindle collider view object being | Touch the music box disc spindle collider view object before and after the music box lid object interaction is complete. | First-person camera moves and rotates towards the music box disc spindle collider view object that simulates the perspective for the music box disc spindle object interaction | First-person camera moves and rotates towards the music box disc spindle collider view object that simulates the perspective for the music box disc spindle object interaction | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | interacted with via touch, when the music box spindle object is visible, and once the music box lid  object interaction is complete. | | when the music box lid object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | when the music box lid object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | 🟩 |
| 27 | First-person camera enters music box disc needle collider object view, upon the music box disc needle collider view object being interacted with via touch, when the music box needle object is visible, and once the music box disc object interaction is complete. | Touch the music box disc needle collider view object before and after the music box disc object interaction is complete. | First-person camera moves and rotates towards the music box disc needle collider view object that simulates the perspective for the music box disc needle object interaction when the music box disc object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | First-person camera moves and rotates towards the music box disc needle collider view object that simulates the perspective for the music box disc needle object interaction when the music box disc object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | 🟩 |
| 28 | First-person camera enters music box overview collider object view, upon the music box overview collider view object being interacted with via touch, when the music box wind up key object is visible, and once the music box disc needle object interaction is complete. | Touch the music box overview collider view object before and after the music box disc needle object interaction is complete. | First-person camera moves and rotates towards the music box overview collider view object that simulates the overviewing perspective for the music box object interaction when the music box disc needle object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | First-person camera moves and rotates towards the music box overview collider view object that simulates the overviewing perspective for the music box object interaction when the music box disc needle object interaction is complete. Otherwise the first-person camera remains at the position and rotation of the music box view object. | 🟩 |
| 29 | First-person camera exits current music box collider object view, upon a swipe down gesture being performed, when the first-person camera is within a music box collider object view. | Swipe down on the devices screen when first-person camera is within a music box collider view objects perspective. | First-person camera moves and rotates towards the music box object view that simulates the perspective for the music box object interaction, when a swipe down gesture interaction is performed within a music box collider object view perspective. | First-person camera moves and rotates towards the music box object view that simulates the perspective for the music box object interaction, when a swipe down gesture interaction is performed within a music box collider object view perspective. | 🟩 |
| 30 | Music box wind up key object can be rotated around music box wind up key object axes, upon first-person camera entering the music box wind up collider object view, and when the active interaction is the music box wind up key objects primary and secondary interactions. | Pinch rotate right or left on the devices screen when first-person camera is within the music box wind up key collider view and music box overview collider view object perspectives. | Music box wind up key object rotates clockwise upon a pinch rotate right gesture interaction being performed, whilst the music box wind up key object rotates anti-clockwise upon a pinch rotate left gesture interaction being performed; when the first-person camera is within the music box wind up key collider view objects perspective. | Music box wind up key object rotates clockwise upon a pinch rotate right gesture interaction being performed, whilst the music box wind up key object rotates anti-clockwise upon a pinch rotate left gesture interaction being performed; when the first-person camera is within the music box wind up key collider view objects perspective. | 🟩 |
| 31 | Music box wind up key objects position can be translated to the music box wind up keyhole objects position, upon the music box wind up key objects rotation | Swipe left on the devices screen when first-person camera is within the music box wind up key collider view objects perspective, and when the music box wind up key object has and has | Music box wind up key objects position is translated to the position of the music box wind up keyhole object, upon a swipe left gesture being performed when the music box wind up key objects | Music box wind up key objects position is translated to the position of the music box wind up keyhole object, upon a swipe left gesture being performed when the music box wind up key objects | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | being aligned with the music box wind up keyhole object, and when being interacted with via swipe left gesture. | not been rotationally aligned with the music box wind up keyhole object. | rotation is aligned with the music box wind up keyhole object. | rotation is aligned with the music box wind up keyhole object. | 🟩 |
| 32 | Music box latch key object can be rotated around music box latch key object axes, upon the first-person camera entering the music box latch key collider object view, and when being interacted with via touch and drag gestures. | Touch and drag on the devices screen when first-person camera is within the music box latch key collider view objects perspective. | Music box latch key object rotates via touch and drag interaction when the first-person camera is within the music box latch key collider view objects perspective. | Music box latch key object rotates via touch and drag interaction when the first-person camera is within the music box latch key collider view objects perspective. | 🟩 |
| 33 | Music box latch key objects position can be translated to the music box latch keyhole objects position, upon the music box latch key objects rotation being aligned with the music box latch keyhole object, and when being interacted with via swipe right gesture. | Swipe right on the devices screen when first-person camera is within the music box latch key collider view objects perspective, and when the music box latch key object has and has not been rotationally aligned with the music box latch keyhole object. | Music box latch key objects position is translated to the position of the music box latch keyhole object, upon a swipe right gesture being performed when the music box latch key objects rotation is aligned with the music box latch keyhole object. | Music box latch key objects position is translated to the position of the music box latch keyhole object, upon a swipe right gesture being performed when the music box latch key objects rotation is aligned with the music box latch keyhole object. | 🟩 |
| 34 | Music box lid object can be rotated around music box lid pivot object axes, upon the first-person camera entering the music box lid collider object view, and when being interacted with via touch and drag gestures. | Touch and drag on the devices screen when first-person camera is within the music box lid collider view objects perspective, and when the music box lid object has and has not been rotationally aligned with the music box lid hinge object. | Music box lid object rotates via touch and drag interaction when the first-person camera is within the music box lid collider view objects perspective. | Music box lid object rotates via touch and drag interaction when the first-person camera is within the music box lid collider view objects perspective. | 🟩 |
| 35 | Music box disc objects position can be translated to the music box disc spindle objects position, upon the first-person camera entering the music box disc spindle collider object view, and when being interacted with via swipe up or held up gestures. | Swipe up or hold up on the devices screen when first-person camera is within the music box disc spindle collider view objects perspective. | Music box disc objects position is translated to the position of the music box disc spindle object, upon a swipe up or held up gesture being performed. | Music box disc objects position is translated to the position of the music box disc spindle object, upon a swipe up or held up gesture being performed. | 🟩 |
| 36 | Music box disc needle object can be rotated around music box disc needle pivot object axes, upon the first-person camera entering the music box disc needle collider object view, and when being | Touch and drag on the devices screen when first-person camera is within the music box disc spindle collider view objects perspective. | Music box disc needle object rotates via touch and drag interaction when the first-person camera is within the music box disc needle collider view objects perspective. | Music box disc needle object rotates via touch and drag interaction when the first-person camera is within the music box disc needle collider view objects perspective. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | interacted with via touch and drag gestures. | | | | 🟩 |
| 37 | First-person camera exits music box wind up key collider object view, upon the primary music box wind up key object interaction being complete. | Swipe left on the devices screen when first-person camera is within the music box wind up key collider view objects perspective, and when the music box wind up key object has been rotationally aligned with the music box wind up keyhole object. | First-person camera exits the music box wind up key collider object view upon the music box wind up key object being inserted into the music box wind up keyhole object. | First-person camera exits the music box wind up key collider object view upon the music box wind up key object being inserted into the music box wind up keyhole object. | 🟩 |
| 38 | First-person camera exits music box latch key collider object view, upon the music box latch key object interaction being complete. | Swipe right on the devices screen when first-person camera is within the music box latch key collider view objects perspective, and when the music box latch key object has and has not been rotationally aligned with the music box latch keyhole object. | First-person camera exits the music box latch key collider object view upon the music box latch key object being inserted into the music box latch keyhole object. | First-person camera exits the music box latch key collider object view upon the music box latch key object being inserted into the music box latch keyhole object. | 🟩 |
| 39 | First-person camera exits music box lid collider object view, upon the music box lid object interaction being complete. | Touch and drag on the devices screen when first-person camera is within the music box lid collider view objects perspective, and when the music box lid object has and has not been rotationally aligned with the music box lid hinge object. | First-person camera exits the music box lid collider object view upon the music box lid object being aligned with the music box lid hinge object. | First-person camera exits the music box lid collider object view upon the music box lid object being aligned with the music box lid hinge object. | Inconsistent working order (not explainable) |
| 40 | First-person camera exits music box disc spindle collider object view, upon the music box disc object interaction being complete. | Swipe up or hold up on the devices screen when first-person camera is within the music box disc spindle collider view objects perspective. | First-person camera exits the music box disc spindle collider object view upon the music box disc object being placed onto the music box disc spindle object. | First-person camera exits the music box disc spindle collider object view upon the music box disc object being placed onto the music box disc spindle object. | 🟩 |
| 41 | First-person camera exits music box disc needle collider object view, upon the music box disc needle interaction being complete. | Touch and drag on the devices screen when first-person camera is within the music box disc spindle collider view objects perspective. | First-person camera exits the music box disc needle collider object view upon the music box disc needle object being aligned with the music box disc spindle object. | First-person camera exits the music box disc needle collider object view upon the music box disc needle object being aligned with the music box disc spindle object. | 🟩 |
| 42 | First-person camera exits music box object view, upon the music box stand and music box object interactions being complete. | Pinch rotate right on the devices screen when first-person camera is within the music box overview collider view perspective. | First-person camera exits the music box object view upon the music box wind up key object finish rotating, and when the music box mirror object transitions to the active animation state. | First-person camera exits the music box object view upon the music box wind up key object finish rotating, and when the music box mirror object transitions to the active animation state. | 🟩 |
| 43 | Music box wardrobe rigid object transitions to the active animation state and emits audio, upon music box stand and music box object interactions being complete, and when music box wardrobe doorknob rigid objects being | Touch the music box wardrobe doorknob rigid objects, upon completing and not completing the music box stand and music box object interactions. | Music box wardrobe rigid object transitions to the active animation state, upon the music box wardrobe doorknob objects being touched, when the music box stand, and music box object interactions are complete. Music box wardrobe object emits wardrobe doors opening sound. | Music box wardrobe rigid object transitions to the active animation state, upon the music box wardrobe doorknob objects being touched, when the music box stand, and music box object interactions are complete. Music box wardrobe object emits wardrobe doors opening sound. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | interacted with, via touch gesture. | | | | 🟩 |
| 44 | First-person camera does not enter music box stand or music box object views, upon the music box stand or music box rigid objects being interacted with via touch, when the music box stand, and music box object interactions are complete. | Touch the music box stand and music box rigid objects, upon the music box stand and music box object interactions being complete. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the music box stand or music box view object perspectives. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the music box stand or music box view object perspectives. | 🟩 |

*Table 14: Black-box test cases, object interaction within puzzle room two, piano puzzle*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | Bookshelf light and light switch objects animate and emit light switch sound, upon being interacted with by touch gesture when the music box puzzle is complete. | Player object traverses towards bookshelf light switch object until relatively close together and touches the bookshelf light switch object. | Bookshelf light and light switch objects animate and emit light switch sound, when the music box puzzle has been complete and when the bookshelf light switch object has been touched. | Bookshelf light and light switch objects animate and emit light switch sound, when the music box puzzle has been complete and when the bookshelf light switch object has been touched. | 🟩 |
| 2 | First-person camera enters bookshelf book object view, upon the bookshelf book object being interacted with via touch and being faced towards, when the bookshelf light has transitioned to the active animation state. | Touch the bookshelf book object when the first-person camera aligns the bookshelf book object in the centre of the FOV and when the bookshelf book object is not centred, upon the bookshelf light transitioning to the active animation state. | First-person camera moves and rotates towards the bookshelf book view object that simulates the perspective for the bookshelf book object interaction when the bookshelf light objects animation state is active. | First-person camera moves and rotates towards the bookshelf book view object that simulates the perspective for the bookshelf book object interaction when the bookshelf light objects animation state is active. | 🟩 |
| 3 | Bookshelf book objects position can be translated to the position of the bookshelf edge, upon the first-person camera entering the bookshelf book view, and when being interacted with via swipe hold down gestures. | Swipe hold down on the devices screen when first-person camera is within the bookshelf book view objects perspective. | Bookshelf book objects position is translated to the position of the bookshelf shelf edge, upon a swipe held down gesture being performed. | Bookshelf book objects position is translated to the position of the bookshelf shelf edge, upon a swipe held down gesture being performed. | 🟩 |
| 4 | Bookshelf book object can be obtained when being interacted with via touch, upon the bookshelf book object being positioned at the bookshelf shelf edge, the bookshelf book object animates as disappearing and emits sound. | Swipe hold down on the devices screen when first-person camera is within the bookshelf book view objects perspective, touch the bookshelf book object when it is positioned at the bookshelf shelf edge. | Bookshelf book objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Bookshelf book objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | First-person camera exits bookshelf book object view, upon the bookshelf book object being obtained. | Swipe hold down on the devices screen when first-person camera is within the bookshelf book view objects perspective, touch the bookshelf book object when it is positioned at the bookshelf shelf edge. | First-person camera exits the bookshelf book object view upon the bookshelf book object being obtained. | First-person camera exits the bookshelf book object view upon the bookshelf book object being obtained. | Inconsistent working order (not explainable) |
| 6 | First-person camera does not enter bookshelf book object view, upon the bookshelf book object being obtained. | Touch where the bookshelf book object was positioned, when the first-person camera is aligned with where the bookshelf book object was positioned in the centre of the FOV. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the bookshelf book view objects perspective. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the bookshelf book view objects perspective. | 🟩 |
| 7 | Book stand hanging light object animates and emits light flickering sounds upon the bookshelf book object being obtained and when the book stand rigid object is being faced towards. | Player object traverses towards book stand rigid object until relatively close together and facing the book stand rigid object. | Book stand hanging light object animates and emits light flickering sounds, when the player is facing the book stand rigid object within close range, and the bookshelf book object has been obtained. | Book stand hanging light object animates and emits light flickering sounds, when the player is facing the book stand rigid object within close range, and the bookshelf book object has been obtained. | 🟩 |
| 8 | First-person camera enters bookstand object view, upon the bookstand object being interacted with via touch and being faced towards, when the bookstand light object has transitioned to the active animation state. | Touch the bookstand object when the first-person camera aligns the bookstand object in the centre of the FOV and when the bookstand object is not centred, upon the bookstand light object transitioning to the active animation state. | First-person camera moves and rotates towards the bookstand view object that simulates the perspective for the bookstand object interaction when the bookstand light objects animation state is active. | First-person camera moves and rotates towards the bookstand view object that simulates the perspective for the bookstand object interaction when the bookstand light objects animation state is active. | 🟩 |
| 9 | Bookshelf book object can be placed upon the first-person camera entering the bookstand object view, and when the bookstand object is being interacted with, via touch. | Touch the bookstand object when within the bookstand view objects perspective. | Bookshelf book object is placed and appears in the scene on top of the bookstand object, upon the bookstand object being touched. | Bookshelf book object is placed and appears in the scene on top of the bookstand object, upon the bookstand object being touched. | 🟩 |
| 10 | Bookshelf book page objects animate upon the bookshelf book object being placed, and when bookshelf book page objects are being interacted with, via swipe left and right gestures. | Swipe left or right on the devices screen when first-person camera is within the bookstand view objects perspective. | Bookshelf book page objects transition to the active animation state upon performing swipe left or right gestures, when within the bookstand view objects perspective. | Bookshelf book page objects transition to the active animation state upon performing swipe left or right gestures, when within the bookstand view objects perspective. | 🟩 |
| 11 | Music sheet objects position can be translated to hovering above the bookshelf book object, upon the bookshelf book page object | Swipe up on the devices screen when first-person camera is within the bookstand view objects perspective, and when the bookshelf book page object has transitioned to | Music sheet objects position is translated to hovering above the bookshelf book object, upon swipe up gestures being performed. | Music sheet objects position is translated to hovering above the bookshelf book object, upon swipe up gestures being performed. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | transitioning to the final active animation state, and when being interacted with, via swipe up gestures. | the final active animation state. | | | 🟩 |
| 12 | Music sheet object can be obtained when being interacted with via touch, upon the music sheet object being positioned above the bookshelf book object, the music sheet object animates as disappearing and emits sound. | Swipe up on the devices screen when first-person camera is within the bookstand view objects perspective, and when the bookshelf book page object has transitioned to the final active animation state, touch the music sheet object when it is positioned above the bookshelf book object. | Music sheet objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Music sheet objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 13 | First-person camera exits bookstand object view, upon the music sheet object being obtained. | Swipe up on the devices screen when first-person camera is within the bookstand view objects perspective, and when the bookshelf book page object has transitioned to the final active animation state, touch the music sheet object when it is positioned above the bookshelf book object. | First-person camera exits the bookstand object view upon the music sheet object being obtained. | First-person camera exits the bookstand object view upon the music sheet object being obtained. | 🟩 |
| 14 | First-person camera does not enter bookstand object view, upon the bookstand and bookshelf book objects being interacted with via touch, when the bookstand and bookshelf book object interactions are complete. | Touch the bookstand and bookshelf book objects, upon the bookstand and bookshelf book object interactions being complete. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the bookstand or bookshelf book object perspectives. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the bookstand or bookshelf book object perspectives. | 🟩 |
| 15 | Piano strobe light objects animates and emit light flashing sounds upon the bookstand and bookshelf book object interactions being complete. | Swipe up on the devices screen when first-person camera is within the bookstand view objects perspective, and when the bookshelf book page object has transitioned to the final active animation state, touch the music sheet object when it is positioned above the bookshelf book object. | Piano strobe light objects animates and emits light flashing sounds, upon the bookstand and bookshelf book interactions being complete. | Piano strobe light objects animates and emits light flashing sounds, upon the bookstand and bookshelf book interactions being complete. | 🟩 |
| 16 | First-person camera enters piano object view, upon the piano rigid object being interacted with via touch and being faced towards, when the piano strobe light objects have transitioned to the active animation states. | Touch the piano rigid object when the first-person camera aligns the piano rigid object in the centre of the FOV and when the piano rigid object is not centred, upon the piano strobe light objects transitioning to the active animation states. | First-person camera moves and rotates towards the piano view object that simulates the perspective for the piano object interactions when the piano strobe light objects animation states are active. | First-person camera moves and rotates towards the piano view object that simulates the perspective for the piano object interactions when the piano strobe light objects animation states are active. | 🟩 |
| 17 | First-person camera enters piano shelf | Touch the piano shelf rigid object when the | First-person camera moves and rotates towards the | First-person camera moves and rotates towards the | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | object view, upon the piano shelf rigid object being interacted with via touch, when the first-person camera has entered the piano object view. | first-person camera has entered and is entering the piano view objects perspective. | piano shelf view object that simulates the perspective for the piano shelf object interaction, when the first-person camera has entered the piano view objects perspective. | piano shelf view object that simulates the perspective for the piano shelf object interaction, when the first-person camera has entered the piano view objects perspective. | 🟩 |
| 18 | Music sheet object can be placed upon the first-person camera entering the piano shelf view, and when the piano shelf view rigid object is being interacted with, via touch. | Touch the piano shelf rigid object when within the piano shelf view objects perspective. | Music sheet object is placed and appears in the scene in front of the piano shelf rigid object, upon the piano shelf rigid object being touched. | Music sheet object is placed and appears in the scene in front of the piano shelf rigid object, upon the piano shelf rigid object being touched. | 🟩 |
| 19 | First-person camera exits piano shelf object view, upon the music sheet object being placed. | Touch the piano shelf rigid object when within the piano shelf view objects perspective. | First-person camera exits the piano shelf object view upon the music sheet object being placed. | First-person camera exits the piano shelf object view upon the music sheet object being placed. | 🟩 |
| 20 | First-person camera enters piano key object view, upon the piano key rigid object being interacted with via touch, and when the music sheet object has been placed. | Touch the piano key rigid object when the music sheet object has and has not been placed, within and when transitioning to and from the piano and piano shelf view object perspectives. | First-person camera moves and rotates towards the piano key view object that simulates the perspective for the piano key object interaction, when the first-person camera has entered the piano view objects perspective, and when the music sheet object has been placed. | First-person camera moves and rotates towards the piano key view object that simulates the perspective for the piano key object interaction, when the first-person camera has entered the piano view objects perspective, and when the music sheet object has been placed. | 🟩 |
| 21 | First-person camera object view can be alternated between the piano shelf and piano key object views, upon entering the piano key object view initially, and when swipe up and down gesture interactions are performed. | Swipe up on the devices screen when first-person camera is within the piano key view objects perspective; swipe down on the devices screen when the first-person camera is within the piano shelf view objects perspective, upon entering the piano key view objects perspective initially. | First-person camera moves and rotates towards the piano key view object that simulates the perspective for the piano key object interaction, upon a swipe down gesture being performed when the first-person camera is within the piano shelf view objects perspective. Vice versa. | First-person camera moves and rotates towards the piano key view object that simulates the perspective for the piano key object interaction, upon a swipe down gesture being performed when the first-person camera is within the piano shelf view objects perspective. Vice versa. | 🟩 |
| 22 | Piano key rigid objects can be played upon the first-person camera entering the piano key object view. Piano key rigid objects animate and emit sound when are being interacted with, via touch. | Touch the piano key rigid objects when within the piano key view objects perspective. | Piano key objects transition to the active animation states and emit sound, upon the piano key rigid objects being touched. Piano key objects cannot be interacted with and played whilst within the active animation state. | Piano key objects transition to the active animation states and emit sound, upon the piano key rigid objects being touched. Piano key objects cannot be interacted with and played whilst within the active animation state. | 🟩 |
| 23 | First-person camera enters piano shelf object view, upon the piano key rigid objects being interacted with via touch, in the wrong sequence, and when the piano key rigid objects are not interacted with for a short period of time, upon a piano | Touch the piano key rigid objects continually and singularly when within the piano key view object perspective. | First-person camera moves and rotates towards the piano shelf view object that simulates the perspective for the music sheet object, when the piano key rigid objects are not interacted with for a short period of time, and when the piano key rigid objects are interacted with continuously in the wrong sequence. | First-person camera moves and rotates towards the piano shelf view object that simulates the perspective for the music sheet object, when the piano key rigid objects are not interacted with for a short period of time, and when the piano key rigid objects are interacted with continuously in the wrong sequence. | 🟩 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | key rigid object being interacted with prior. |  |  |  | 🟩 |
| 24 | First-person camera exits piano object view, upon the piano puzzle interaction being complete. | Touch the piano key rigid objects continually in the correct sequence when within the piano key view object perspective. | First-person camera exits the piano object view upon the piano puzzle interaction being complete. | First-person camera exits the piano object view upon the piano puzzle interaction being complete. | 🟩 |
| 25 | First-person camera does not enter piano, piano shelf, and piano key object views, upon the piano, piano shelf and piano key objects being interacted with, via touch, and when the piano object interactions are complete. | Touch the piano, piano shelf, and piano key objects, upon the piano object interactions being complete. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the piano, piano shelf, and piano key view object perspectives. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the piano, piano shelf, and piano key view object perspectives. | 🟩 |

*Table 15: Black-box test cases, object interaction within puzzle room one, telephone puzzle*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | First-person camera enters telephone rigid object view, upon the telephone rigid object being interacted with via touch and being faced towards. | Touch the telephone rigid object when the first-person camera aligns the telephone rigid object in the centre of the FOV and when the telephone rigid object is not centred. | First-person camera moves and rotates towards the telephone view object that simulates the perspective for the telephone object interactions. | First-person camera moves and rotates towards the telephone view object that simulates the perspective for the telephone object interactions. | 🟩 |
| 2 | Telephone earpiece object animates and emits sounds upon the first-person camera entering the telephone rigid object view, when a swipe hold up gesture is performed, and when the telephone ringing sound is being played. | Swipe hold up on the devices screen when first-person camera is within the telephone view objects perspective. | Telephone earpiece object transitions to the active animation states and emits sound, upon a swipe held up gesture being performed when within the telephone view objects perspective. Telephone object ringing sound stops playing. | Telephone earpiece object transitions to the active animation states and emits sound, upon a swipe held up gesture being performed when within the telephone view objects perspective. Telephone object ringing sound stops playing. | 🟩 |
| 3 | First-person camera exits telephone object view, upon the telephone earpiece object transitioning to the static animation state. Telephone earpiece object emits sound. | Swipe hold up on the devices screen when first-person camera is within the telephone view objects perspective. | First-person camera exits the telephone object view upon the telephone earpiece object transitioning to the static animation state, telephone earpiece object emits sound. | First-person camera exits the telephone object view upon the telephone earpiece object transitioning to the static animation state, telephone earpiece object emits sound. | 🟩 |
| 4 | First-person camera does not enter telephone object view, upon the telephone earpiece object being initially interacted with, via touch, and until the clipboard note has been obtained. | Touch the telephone object, upon the initial telephone earpiece interaction being complete. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the telephone view objects perspective. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the telephone view objects perspective. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | Ornament stand hanging light objects animate and emit light flickering sounds upon the telephone objects initial interaction being complete, and when the ornament stand rigid objects are being faced towards. | Player object traverses towards ornament stand rigid objects until relatively close together and facing the ornament stand rigid objects. | Ornament stand hanging light objects animate and emit light flickering sounds, when the player is facing the ornament stand rigid objects within close range, and the initial telephone object interaction has been complete. | Ornament stand hanging light objects animate and emit light flickering sounds, when the player is facing the ornament stand rigid objects within close range, and the initial telephone object interaction has been complete. | 🟩 |
| 6 | First-person camera enters face mask rigid object view, upon the ornament stand light objects transitioning to the active animation state, and when the face mask rigid object is being faced towards, from a similar angular offset as the first-person camera. | Touch the face mask rigid object when the first-person camera aligns the face mask rigid object in the centre of the FOV and when the face mask rigid object is not centred. As well as when the first-person camera does and does not share a similar angular offset to the face mask rigid object. | First-person camera moves and rotates towards the face mask view object that simulates the perspective for the face mask object. First-person camera does not move and rotate towards the face mask view object when the audio device objects audio sequence is playing. | First-person camera moves and rotates towards the face mask view object that simulates the perspective for the face mask object. First-person camera does not move and rotate towards the face mask view object when the audio device objects audio sequence is playing. | 🟩 |
| 7 | First-person camera exits face mask object view, upon the face mask object interaction being complete. | Touch the face mask rigid object when the first-person camera aligns the face mask rigid object in the centre of the FOV and when the face mask rigid object is not centred. As well as when the first-person camera does and does not share a similar angular offset to the face mask rigid object. | First-person camera exits the face mask object view upon the face mask object interaction being complete. | First-person camera exits the face mask object view upon the face mask object interaction being complete. | Inconsistent working order (not explainable) |
| 8 | Glass cabinet window rigid object animates and emits sound upon the ornament stand light objects transitioning to the active animation state, and when the glass cabinet window object is being interacted with, via touch. | Touch the glass cabinet window rigid object, upon the ornament stand light objects transitioning to the active animation state. | Glass cabinet window object transitions to the active animation state and emits sound, upon being touched. | Glass cabinet window object transitions to the active animation state and emits sound, upon being touched. | 🟩 |
| 9 | Clipboard note object can be obtained when being interacted with via touch, upon the ornament stand light objects transitioning to the active animation state, the music sheet object animates as disappearing and emits sound. | Player object traverses towards clipboard note object until relatively close together and touches the clipboard note object. | Clipboard note objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | Clipboard note objects scale reduces to being invisible upon being touched, where the item obtained sound is then played to signal its possession. | 🟩 |
| 10 | First-person camera enters clipboard rigid object view, upon the clipboard rigid object being interacted with, via | Touch the clipboard rigid object when the clipboard note object has been obtained and when the clipboard note has been placed. | First-person camera moves and rotates towards the clipboard view object that simulates the perspective for the clipboard note object interaction. First- | First-person camera moves and rotates towards the clipboard view object that simulates the perspective for the clipboard note object interaction. First- | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | touch, when the clipboard note object has been obtained or placed. | | person camera does not move and rotate towards the clipboard view object when the audio device objects audio sequence is playing. | person camera does not move and rotate towards the clipboard view object when the audio device objects audio sequence is playing. | 🟩 |
| 11 | Clipboard note object can be placed upon the first-person camera entering the clipboard object view, when the clipboard note object has not been placed prior. | Touch the clipboard rigid object when within the clipboard view objects perspective, when the clipboard note object has and has not been placed prior. | Clipboard note object is placed and appears on top of the clipboard object, upon the clipboard object being touched, when the clipboard note object has not been placed prior. | Clipboard note object is placed and appears on top of the clipboard object, upon the clipboard object being touched, when the clipboard note object has not been placed prior. | 🟩 |
| 12 | First-person camera exits clipboard object view, upon the clipboard note object being placed or after a short period of time. | Touch the clipboard rigid object when within the clipboard view objects perspective. | First-person camera exits the clipboard object view, upon the clipboard note object being placed or after a short period of time when the clipboard note object has been placed prior. | First-person camera exits the clipboard object view, upon the clipboard note object being placed or after a short period of time when the clipboard note object has been placed prior. | 🟩 |
| 13 | Audio device rigid object can be played upon the ornament stand light objects transitioning to the active animation state, and when the audio device is being interacted with, via touch. | Touch the audio device rigid object when the ornament stand light objects enter the active animation state. | Audio device object emits Morse code sound sequence, upon the audio device object being touched, and when the ornament stand light objects enter the active animation state. Audio device objects sound sequence cannot be played again, until current sound sequence iteration has finished playing. | Audio device object emits Morse code sound sequence, upon the audio device object being touched, and when the ornament stand light objects enter the active animation state. Audio device objects sound sequence cannot be played again, until current sound sequence iteration has finished playing. | 🟩 |
| 14 | First-person camera enters Morse code book rigid object view, upon the ornament stand light objects transitioning to the active animation state, and when the Morse code book rigid object is being interacted with, via touch. | Touch the Morse code rigid object when the ornament stand light objects enter the active animation state. | First-person camera moves and rotates towards the Morse code book view object that simulates the perspective for the Morse code book page object. | First-person camera moves and rotates towards the Morse code book view object that simulates the perspective for the Morse code book page object. | 🟩 |
| 15 | First-person camera exits Morse code book object view, after a short period of time. | Touch the Morse code rigid object when the ornament stand light objects enter the active animation state. | First-person camera exits the Morse code object view, after a short period of time. | First-person camera exits the Morse code object view, after a short period of time. | 🟩 |
| 16 | First-person camera enters telephone rigid object view, upon the telephone rigid object being interacted with via touch and being faced towards, and when the clipboard note rigid object has been placed. | Touch the telephone rigid object when the first-person camera aligns the telephone rigid object in the centre of the FOV and when the telephone rigid object is not centred. | First-person camera moves and rotates towards the telephone view object that simulates the perspective for the telephone object interactions. First-person camera does not move and rotate towards the clipboard view object when the audio device objects audio sequence is playing. | First-person camera moves and rotates towards the telephone view object that simulates the perspective for the telephone object interactions. First-person camera does not move and rotate towards the clipboard view object when the audio device objects audio sequence is playing. | 🟩 |
| 17 | Telephone button rigid objects can be pressed upon the first-person camera entering the telephone rigid | Touch the telephone button rigid objects when within the telephone view objects perspective. | Telephone button objects transition to the active animation states and emit sound, upon the telephone button rigid objects being touched. Telephone button | Telephone button objects transition to the active animation states and emit sound, upon the telephone button rigid objects being touched. Telephone button | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | object view. Telephone button rigid objects animate and emit sound when are being interacted with, via touch. | | objects cannot be interacted with and played whilst within the active animation state. | objects cannot be interacted with and played whilst within the active animation state. | 🟩 |
| 18 | Telephone number object displays the telephone button rigid object input, upon the telephone button rigid objects being interacted with, via touch. Telephone number object resets the telephone button rigid object input, when the telephone number object exceeds the limit for displaying input. | Touch the telephone button rigid objects when within the telephone view objects perspective. | Telephone number object displays the telephone button rigid object input, upon the telephone button rigid objects being interacted with, via touch. Telephone number object resets the telephone button rigid object input, when the telephone number object exceeds the limit for displaying input. | Telephone number object displays the telephone button rigid object input, upon the telephone button rigid objects being interacted with, via touch. Telephone number object resets the telephone button rigid object input, when the telephone number object exceeds the limit for displaying input. | 🟩 |
| 19 | Telephone earpiece object animates and emits sounds upon the first-person camera entering the telephone rigid object view, when a swipe hold up gesture is performed, and when the telephone number object is displaying the limit for telephone button rigid object input. | Swipe hold up on the devices screen when first-person camera is within the telephone view objects perspective. | Telephone earpiece object transitions to the active animation states and emits sound, upon a swipe held up gesture being performed when within the telephone view objects perspective. | Telephone earpiece object transitions to the active animation states and emits sound, upon a swipe held up gesture being performed when within the telephone view objects perspective. | 🟩 |
| 20 | First-person camera exits telephone rigid object view, upon a swipe hold down gesture being performed, when the first-person camera is within the telephone rigid object view. | Swipe hold down on the devices screen when first-person camera is within a telephone rigid view objects perspective. | First-person camera exits the face mask object view upon a swipe hold down gesture interaction being performed when within the telephone object view. | First-person camera exits the face mask object view upon a swipe hold down gesture interaction being performed when within the telephone object view. | 🟩 |
| 21 | First-person camera exits telephone rigid object view, upon the telephone rigid object interaction being complete. | Touch the telephone button rigid objects when within the telephone view objects perspective, press the telephone button rigid objects in the correct sequence, and perform a swipe hold up gesture upon the telephone number object input reaching the limit. | First-person camera exits the telephone object view upon the telephone object interaction being complete. | First-person camera exits the telephone object view upon the telephone object interaction being complete. | Inconsistent working order (not explainable) |
| 22 | First-person camera does not enter face mask, clipboard, Morse code book and telephone object views, upon the face mask, clipboard, Morse code book and telephone objects being interacted | Touch the face mask, clipboard, Morse code book and telephone objects, upon the telephone object interaction being complete. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the face mask, clipboard, Morse code book and telephone view object perspectives. | First-person camera remains at the position and rotation of the player object. First-person camera movement does not resemble attempts to enter the face mask, clipboard, Morse code book and telephone view object perspectives. | 🟩 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | with, via touch, and when the telephone object interaction is complete. |  |  |  | 🟩 |

*Table 16: Black-box test cases, scene transitioning*

| Case | Summary | Process | Expected result(s) | Actual result(s) | Passed? |
|---|---|---|---|---|---|
| 1 | First level scene is transitioned to and loaded, upon the loading screen scene start button and disclaimer window button objects being interacted with, via touch. | Touch the disclaimer window button object, upon the start button object being touched and disclaimer window button object appearing. | First level scene is transitioned to and loaded, upon the loading screen scene start button and disclaimer window button objects being touched. | First level scene is transitioned to and loaded, upon the loading screen scene start button and disclaimer window button objects being touched. | 🟩 |
| 2 | First puzzle room scene is transitioned to and loaded, upon the first level scene being loaded, and when the player object collides with the first puzzle room door and level trigger objects. As well as when the first puzzle room interactions have not been complete. | Player object traverses towards level one door object until relatively close together and collides with the level one door trigger object, where the level one door object transitions to the active animation state. Upon the level one door object entering the active animation state, the player object traverses towards the level one door object again, until relatively close together and collides with the level one trigger object. | First puzzle room scene is transitioned to and loaded, upon the player object colliding with the level one trigger object, and when the first puzzle room interactions have not been complete. | First puzzle room scene is transitioned to and loaded, upon the player object colliding with the level one trigger object, and when the first puzzle room interactions have not been complete. | 🟩 |
| 3 | Second puzzle room scene is transitioned to and loaded, upon the first level scene being loaded, and when the player object collides with the second puzzle room door and level trigger objects. As well as when the second puzzle room interactions have not been complete, whilst the first puzzle room interactions have been complete. | Player object traverses towards level two door object until relatively close together and collides with the level two door trigger object, where the level two door object transitions to the active animation state. Upon the level two door object entering the active animation state, the player object traverses towards the level two door object again, until relatively close together and collides with the level two trigger object. | Second puzzle room scene is transitioned to and loaded, upon the player object colliding with the level two trigger object, and when the second puzzle room interactions have not been complete, but the first puzzle room interactions have been complete. | Second puzzle room scene is transitioned to and loaded, upon the player object colliding with the level two trigger object, and when the second puzzle room interactions have not been complete, but the first puzzle room interactions have been complete. | 🟩 |
| 4 | Third puzzle room scene is transitioned to and loaded, upon the first level scene being loaded, and when the player object collides with the third puzzle room door and level trigger objects. As well as when the third puzzle room interactions have not been complete, whilst the first and second puzzle room | Player object traverses towards level three door object until relatively close together and collides with the level three door trigger object, where the level three door object transitions to the active animation state. Upon the level three door object entering the active animation state, the player object traverses towards the level three door object again, until relatively close together and collides with | Third puzzle room scene is transitioned to and loaded, upon the player object colliding with the level three trigger object, and when the third puzzle room interactions have not been complete, but the first and second puzzle room interactions have been complete. | Third puzzle room scene is transitioned to and loaded, upon the player object colliding with the level three trigger object, and when the third puzzle room interactions have not been complete, but the first and second puzzle room interactions have been complete. | 🟩 |

| | | | | | |
|---|---|---|---|---|---|
| | interactions have been complete. | the level three trigger object. | | | 🟩 |
| 5 | First level scene is transitioned to and loaded, upon the player object colliding with the first puzzle room door level and leave level one trigger objects, when the first puzzle room interactions are complete. | Player object traverses towards puzzle room one door object until relatively close together and collides with the leave level one trigger object, where the puzzle room one door object transitions to the active animation state. Upon the puzzle room one door object entering the active animation state, the player object traverses towards the puzzle room one door object again, until relatively close together and collides with the leave level one trigger object. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level one trigger object, when the first puzzle room interactions have been complete. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level one trigger object, when the first puzzle room interactions have been complete. | 🟩 |
| 6 | First level scene is transitioned to and loaded, upon the player object colliding with the second puzzle room door level and leave level one trigger objects, when the second puzzle room interactions are complete. | Player object traverses towards puzzle room two door object until relatively close together and collides with the leave level two trigger object, where the puzzle room two door object transitions to the active animation state. Upon the puzzle room two door object entering the active animation state, the player object traverses towards the puzzle room two door object again, until relatively close together and collides with the leave level two trigger object. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level two trigger object, when the second puzzle room interactions have been complete. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level two trigger object, when the second puzzle room interactions have been complete. | 🟩 |
| 7 | First level scene is transitioned to and loaded, upon the player object colliding with the third puzzle room door level and leave level three trigger objects, when the third puzzle room interactions are complete. | Player object traverses towards puzzle room three door object until relatively close together and collides with the leave level three trigger object, where the puzzle room three door object transitions to the active animation state. Upon the puzzle room three door object entering the active animation state, the player object traverses towards the puzzle room three door object again, until relatively close together and collides with the leave level three trigger object. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level three trigger object, when the third puzzle room interactions have been complete. | First level scene is transitioned to and loaded, upon the player object colliding with the leave level three trigger object, when the third puzzle room interactions have been complete. | 🟩 |

## *Appendix P:*

*Table 17: Performance profiling test cases, loading screen scene*

| Case | Summary | Process | Actual result(s) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | Loading screen scene start-up. | Load into the loading screen scene, capture performance using Unity profiler. | Peak usage: CPU (66.8%), GPU (79.7%), RAM (0.414GB)<br>Minimum usage: CPU (40.5%), GPU (20.2%), RAM (0.412GB)<br>Average FPS: 142.80 |
| 2 | Loading screen scene disclaimer window button appear. | Load into the loading screen scene, interact with start button object and capture performance using Unity profiler. | Peak usage: CPU (53.0%), GPU (79.6%), RAM (0.418GB)<br>Minimum usage: CPU (33.1%), GPU (9.1%), RAM (0.417GB)<br>Average FPS: 142.07 |
| 3 | Loading screen scene disclaimer window button disappear. | Load into the loading screen scene, interact with disclaimer window button object and capture performance using Unity profiler. | Peak usage: CPU (64.6%), GPU (92.2%), RAM (0.454GB)<br>Minimum usage: CPU (37.1%), GPU (41.9%), RAM (0.451GB)<br>Average FPS: 142.81 |

*Table 18: Performance profiling test cases, first level scene*

| Case | Summary | Process | Actual result(s) |
|---|---|---|---|
| 1 | First level scene start-up. | Load into the first level scene, capture performance using Unity profiler. | Peak usage: CPU (51.7%), GPU (96.6%). RAM (1.54GB)<br>Minimum usage: CPU (96.2%), GPU (6.1%), RAM (1.54GB)<br>Average FPS: 140.23 |
| 2 | First level scene roaming. | Load into the loading screen scene, roam the scene and capture performance using Unity profiler. | Peak usage: CPU (72.8%), GPU (80.2%), RAM (1.85GB)<br>Minimum usage: CPU (32.7%), GPU (3.2%), RAM (1.84GB)<br>Average FPS: 116.44 |
| 3 | First level scene puzzle door objects transition to the active animation state. | Load into the loading screen scene, collide with the puzzle door object triggers and capture performance using Unity profiler. | Peak usage: CPU (80.1%), GPU (96.1%), RAM (1.82GB)<br>Minimum usage: CPU (45.4%), GPU (4.8%), RAM (1.82GB)<br>Average FPS: 140.38 |

*Table 19: Performance profiling test cases, puzzle room two scene*

| Case | Summary | Process | Actual result(s) |
|---|---|---|---|
| 1 | Second puzzle room start-up. | Load into the second puzzle room scene, capture performance using Unity profiler. | Peak usage: CPU (69.6%), GPU (79.0%), RAM (0.81GB)<br>Minimum usage: CPU (29.2%), GPU (32.9%), RAM (0.80GB)<br>Average FPS: 124.67 |
| 2 | Second puzzle room roaming. | Load into the second puzzle room scene, roam the scene and capture performance using Unity profiler. | Peak usage: CPU (75.4%), GPU (89.1%), RAM (0.78GB)<br>Minimum usage: CPU (49.2%), GPU (10.8%), RAM (0.77GB)<br>Average FPS: 128.45 |
| 3 | Second puzzle room safe puzzle interaction. | Interact with the safe object and capture performance using Unity profiler. | Peak usage: CPU (72.3%), GPU (67.1%), RAM (0.77GB)<br>Minimum usage: CPU (51.6%), GPU (38.7%), RAM (0.77GB)<br>Average FPS: 137.56 |
| 4 | Second puzzle room table piece puzzle interaction. | Interact with the table globe object and capture performance using Unity profiler. | Peak usage: CPU (99.7%), GPU (85.3%), RAM (0.79GB)<br>Minimum usage: CPU (50.3%), GPU (3.1%), RAM (0.79GB)<br>Average FPS: 137.99 |
| 5 | Second puzzle room music box picture interaction. | Interact with the music box picture object and capture performance using Unity profiler. | Peak usage: CPU (76.0%), GPU (67.0%), RAM (0.79GB)<br>Minimum usage: CPU (14.3%), GPU (28.9%), RAM (0.79GB)<br>Average FPS: 139.02 |
| 6 | Second puzzle room carousel music box interaction. | Interact with the carousel music box object and capture performance using Unity profiler. | Peak usage: CPU (79.2%), GPU (90.3%), RAM (0.84GB)<br>Minimum usage: CPU (51.4%), GPU (15.4%), RAM (0.84GB)<br>Average FPS: 132.75 |

| | | | |
|---|---|---|---|
| 7 | Second puzzle room music box interaction. | Interact with the music box object and capture performance using Unity profiler. | Peak usage: CPU (70.7%), GPU (92.9%), RAM (0.97GB)<br>Minimum usage: CPU (49.7%), GPU (3.1%), RAM (0.97GB)<br>Average FPS: 108.61 |
| 8 | Second puzzle room wardrobe interaction. | Interact with the wardrobe object and capture performance using Unity profiler. | Peak usage: CPU (65.9%), GPU (86.9%), RAM (1.3GB)<br>Minimum usage: CPU (45.1%), GPU (22.8%), RAM (1.3GB)<br>Average FPS: 106.91 |
| 9 | Second puzzle room bookshelf book interaction. | Interact with the bookshelf book object and capture performance using Unity profiler. | Peak usage: CPU (72.1%), GPU (73.5%), RAM (1.26GB)<br>Minimum usage: CPU (50.5%), GPU (18.0%), RAM (1.26GB)<br>Average FPS: 136.51 |
| 10 | Second puzzle room bookstand interaction. | Interact with the bookstand object and capture performance using Unity profiler. | Peak usage: CPU (75.0%), GPU (96.7%), RAM (1.27GB)<br>Minimum usage: CPU (34.9%), GPU (9.4%), RAM (1.27GB)<br>Average FPS: 129.57 |
| 11 | Second puzzle room piano interaction. | Interact with the piano object and capture performance using Unity profiler. | Peak usage: CPU (76.3%), GPU (71.4%), RAM (1.3GB)<br>Minimum usage: CPU (50.8%), GPU (13.6%), RAM (1.3GB)<br>Average FPS: 136.37 |

*Table 20: Performance profiling test cases, puzzle room one scene*

| Case | Summary | Process | Actual result(s) |
|---|---|---|---|
| 1 | First puzzle room start-up. | Load into the first puzzle room scene, capture performance using Unity profiler. | Peak usage: CPU (74.5%), GPU (94.6%), RAM (1.22GB)<br>Minimum usage: CPU (48.7%), GPU (26.9%), RAM (1.21GB)<br>Average FPS: 140.30 |
| 2 | First puzzle room roaming. | Load into the first puzzle room scene, roam the scene and capture performance using Unity profiler. | Peak usage: CPU (74.7%), GPU (89.5%), RAM (1.22GB)<br>Minimum usage: CPU (45.4%), GPU (6.6%), RAM (1.21GB)<br>Average FPS: 135.48 |
| 3 | First puzzle room initial telephone interaction. | Interact with the telephone object and capture performance using Unity profiler. | Peak usage: CPU (64.8%), GPU (74.0%), RAM (1.21GB)<br>Minimum usage: CPU (30.1%), GPU (28.1%), RAM (1.21GB)<br>Average FPS: 141.26 |
| 4 | First puzzle room face mask interaction. | Interact with the face mask object and capture performance using Unity profiler. | Peak usage: CPU (78.0%), GPU (83.8.0%), RAM (1.22GB)<br>Minimum usage: CPU (46.5%), GPU (14.6%), RAM (1.22GB)<br>Average FPS: 139.76 |
| 5 | First puzzle room glass cabinet interaction. | Interact with the glass cabinet object and capture performance using Unity profiler. | Peak usage: CPU (78.0%), GPU (94.4%), RAM (1.24GB)<br>Minimum usage: CPU (46.5%), GPU (33.6%), RAM (1.23GB)<br>Average FPS: 140.74 |
| 6 | First puzzle room clipboard interaction. | Interact with the clipboard object and capture performance using Unity profiler. | Peak usage: CPU (71.9%), GPU (79.3%), RAM (1.25GB)<br>Minimum usage: CPU (46.1%), GPU (48.7%), RAM (1.24GB)<br>Average FPS: 139.57 |
| 7 | First puzzle room audio device interaction. | Interact with the audio device object and capture performance using Unity profiler. | Peak usage: CPU (76.3%), GPU (94.8%), RAM (1.28GB)<br>Minimum usage: CPU (31.7%), GPU (45.5%), RAM (1.28GB)<br>Average FPS: 139.57 |
| 8 | First puzzle room telephone puzzle interaction. | Interact with the telephone object and capture performance using Unity profiler. | Peak usage: CPU (74.9%), GPU (76.0%), RAM (1.28GB)<br>Minimum usage: CPU (46.7%), GPU (3.4%), RAM (1.27GB)<br>Average FPS: 142.09 |