Abstract

This is an experience designed to introduce and engage middle or high school students to a probability topic in Mathematics. It could also be used by anyone with an interest in probability, particularly those that would like to experience the Monty Hall probability scenario.

MonTy Hall: Probability  
TECHNICAL DESIGN DOCUMENT

Developer: <Sion Harman>

Contents

[About 3](#_Toc51679645)

[Change Log 4](#_Toc51679646)

[Team Members 4](#_Toc51679647)

[Development Environment 5](#_Toc51679648)

[Software Requirements 5](#_Toc51679649)

[Accounts 6](#_Toc51679650)

[Third Party Libraries 7](#_Toc51679651)

[Version Control 8](#_Toc51679652)

[Repository 8](#_Toc51679653)

[Contributors 8](#_Toc51679654)

[Commit Message Format: 8](#_Toc51679655)

[Target Platform 9](#_Toc51679656)

[<Platform> 9](#_Toc51679657)

[<Platform> Limitations 9](#_Toc51679658)

[Minimum <Platform> Specs 9](#_Toc51679659)

[Release Build Instructions 9](#_Toc51679660)

[Deliverables 9](#_Toc51679661)

[Controls 10](#_Toc51679662)

[Keyboard / Mouse 10](#_Toc51679663)

[Controller 10](#_Toc51679664)

[Mobile / Touch 10](#_Toc51679665)

[Custom Game Systems 10](#_Toc51679666)

[Coding Standards 10](#_Toc51679667)

[Coding Standards Enforcement 10](#_Toc51679668)

[Technical Goals and Challenges 11](#_Toc51679669)

[Technical Goals: 11](#_Toc51679670)

[Technical Risks: 11](#_Toc51679671)

[Risk Avoidance 11](#_Toc51679672)

# About

Describe the project / Game. Focus on it as a product, rather than a gameplay pitch. (1 paragraph)

This is an experience designed to introduce and engage middle or high school students to a probability topic in Mathematics. It could also be used by anyone with an interest in probability, particularly those that would like to experience the Monty Hall probability scenario.

Describe the purpose of this document (1 paragraph)

This document will outline the technical parts of the game/experience including software requirements and development environment details. Also included are game overviews and gameplay systems. This document is designed for anyone who is involved with the development of the experience, including asset information, GUI, programming, and development.

# Change Log

Updates made to the document should be described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date of change | Description |
| 0.0.1 | Harman | 15/3/2023 | Initial Template created |
| 0.0.2 | Harman | 20/3/23 | Filled in details and created flow charts for scripts 1 and 2 |
| 0.0.3 | Harman | 30/3/23 | Added flowchart for simulation game state and received feedback on pseudocode and flowcharts |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Development Environment

This section outlines the required software and systems required for development of this project.

## Software Requirements

The below table outlines the software requirements for development of this project. Developers contributing to the project are required to use the approved software outlined below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Version | License | Used By | Used For |
| Unity 3D | 2022.2.10f1 | Education | Programmers, Designers, Artists | Development of Game |
| Unity 3D | 2022.2.10f1 | Free | Programmers, Designers, Artists  (At Home) | Development of Game |
| Visual Studio 2022 | 2022 | Free | Programmers | Development of Game |
| Web Browser | - | Free | All | Git repository |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Libraries

Unity/Unreal comes with a default collection of plugins, tools and assets. Its plausible, and often encouraged to pull in additional assets, tools, plugins or scripts etc. developed by a 3rd party. Identify both engine and system libraries used in the project, and especially any 3rd party ones used, including licensing information on its usage.

|  |  |  |
| --- | --- | --- |
| Asset/Library/Package name | License | Used For |
| Input system | Unity | First person controller default in Unity |
| Probuilder |  | Building walls around the doors |
| Asset “ARCADE: FREE Racing Car” | Unity Store | Spawn car behind winning door |
| Asset “Easy Primitive Animals – Farm Edition ” | Unity Store | Spawn sheep behind opened door |
|  |  |  |

## Version Control

### Repository

### <https://github.com/sib0rg-au/Monty_Hall_V2>

### Contributors

* <sib0rg-au>

## Commit Message Format

Standard commit message will include the following information:

Please update, add or remove any details that you identify as important to include in commit message.

* **Type**: Represents the type of change, often the “Type” can be inferred based on the associated ticket in your project management tool, which may include: FIX, FEATURE, REFACTOR, DOC, TEST etc.
* **Scope:** Refers to the area of the project being changed, could refer to things like (menu) (inventory) (save\_system) (level) (controls) etc. Scopes may change throughout development but can be broadly identified. Outline the scopes below that seem suitable for your project
  + MENU
  + LEVEL
  + **…**
* **TaskId:** Id of the associated ticket representing the change.
* **Summary:** A short description of what has been changed.

**Format:**

|  |
| --- |
| Game (student assessment) : TaskId : Monty Hall Probability Game |

**Examples:**

|  |
| --- |
| Feature (menu) : #1302 : Added Exit button to main menu |
| Fix (menu) : #1395 : Updated button prefab with so that hover works on web builds |
| Feature (sandbox) : #1129 : Added rock asset to test scene, Created Rock prefab |
| <add your own examples> |
| <add your own examples> |
| <add your own examples> |

# Game Overview

## Description

This is a general description of gameplay. It could be taken straight from the GDD.

## This is a simple probability exercise to get people thinking about probability in the form of a game/gameshow.

Gameplay has the player selecting a prize, hidden behind one of 3 doors.

Once a door is selected, one of the other 2 doors is removed from the scenario, one that does not have the prize behind it.

The player is then given another choice before the prize door is revealed: stick with their initial door choice, or swap to the last remaining door.

The prize is then revealed behind the correct door.

This can then be reset and repeated.

On the 3rd play through, narration is played, explaining the Monty Hall paradox and it’s history, along with probabilities.

## Genre

The game experience is an educational one in a gameshow type setting.

## Perspective

Identify the basic perspective of play through the game (e.g top-down, First-person, Third-person, isometric, 2D side-scrolling, etc). This helps provide context for what is being displayed.

This is a first person game to increase engagement and sense of scale. Potential for a VR title

## Target Platforms

This project will be deployed to the following platforms:

* WebGL / Browser

Duplicate this section for each desired platform

### <Platform> Limitations

Outline <platform> limitations, provide short description of the limitation. could include:

* Available inputs (keyboard, mouse, touch, controllers etc)
* Performance constraints (max number of particles, game objects etc)

### Minimum <Platform> Specs

Outline the expected minimum system requirements required to run the project in release build.  
The minimum/maximum specs should consider target audience system specs and drive both technical and non-technical design decisions to ensure project runs on specified devices.

* System spec
* System spec

### WebGL/Browser Limitations

Limitations include system specs of the user. Also being web-based, users may expect a very fast load time to run the game.

* Inputs will depend on device. Will base development on computer inputs initially (keyboard, mouse) and add touch and controllers later for mobile support.
* Performance constraints (There won’t be a lot of particle effects and this will take place in a small outdoor environment with a skybox. Aiming for maximum of 50 objects, but will likely be much less)

### Minimum <Platform> Specs

Minimum specs TBD.

## Feature List

The project’s features include:

This is just a list – Details of feature implementation come in later sections.

* Feature 1
* Feature 2
* Feature 3
* Etc
* <PlayerSimulation mode (see below)>Interactable Buttons (x5): x3 for doors, x1 for reset, x1 for peeking at answer before progressing.
* <ScaleSimulation mode (see below)> Interactable buttons (X4): select simulation number, select choice (swap or stay), run simulation, reset data
* Doors that can open. Non-player controlled.
* A narrator character.

# Game Flow & Structure

This section of the document outlines the high-level structure and order of play for the project.

Game Modes & Handling  
This is a technical description of how modes are handled in the project. At what point are modes selected (if at all)? What scripts contain the game mode information, etc? Still cover this even if you have only one mode, as there will always effectively be ‘game mode’ information executed.

* Mode handling point 1
* Mode handling point 2
* Etc.
* Menu screen
* Run personal simulation
* Run scale simulation

Game Mode - <MenuScreen>   
This section repeats for every mode in the game.

Description   
Basic concise description of how the game mode works.  
This is the initial screen a player is met with. It contains interactable buttons to start running the personal or scale simulation, and control options.

Scale simulation can only be run if the player has gone through 3 rounds of personal simulation

Objectives   
These should ideally be actual objectives in the game mode, rather than general ones like “earn points”. We are defining technical criteria for objectives, not ‘good play’.

Get the user to select required game state.

Objective Tracking   
Describe how objectives will be handled technically in this mode. If it is handled by the event handler, simply make that clear and reference that.

When a Menu option button is selected, the player will be taken to the required game scene.

Game Mode - <PersonalSimulation> 

Description   
This is the active game state where the player plays the game by choosing doors and listens to the host character instructions.

Player chooses a button relating to a door

They are then presented with a choice to keep their choice or swap given the new scenario.

They are then presented with the door which had the prize behind it.

This can then loop back and reset by hitting the reset button or after 10 seconds has expired.

Objectives 

On start, let the user know how to play the game (host audio)

Get the user to select and interact with door buttons.

Get the user to understand the probability of their choices on the 3rd round.

Objective Tracking 

1. When a button selected, play game host audio, and remove a door that doesn’t have the prize or is selected by the player.
2. When second button selected (stay or swap), open winning door, revealing car prize. Play host audio.
3. On the 3rd round, host audio informs the player of the probabilities related to the game.

Game Mode - <ScaleSimulation> 

Description   
This is the active game state where the player runs simulations for the Monty Hall game to see the statistical probability of staying or swapping over many simulations.

The player interacts with objects to choose how many times they would like to run the simulation and if the choice is keep or swap.

Player can then view the statistics for winning when they swap vs when they stay with their choice.

Objectives 

On start, play host instruction audio

Get the user to select simulation number.

Get the user to select choice of ‘stay’ or ‘swap’.

Show user a 3D percentage graph as simulation runs.

Objective Tracking 

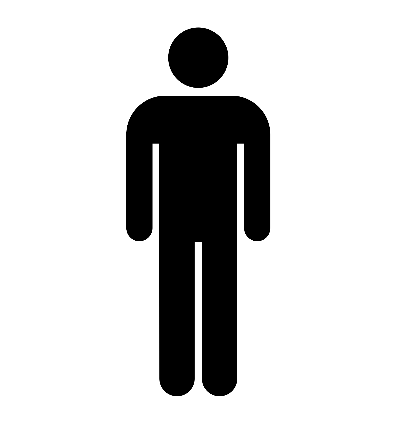
1. Save the number of simulations
2. Save the stay or swap choice
3. Run the simulation and instantiate objects for every winning round?

Mission / Level Structure  
This section identifies the structure of gameplay across levels / missions.

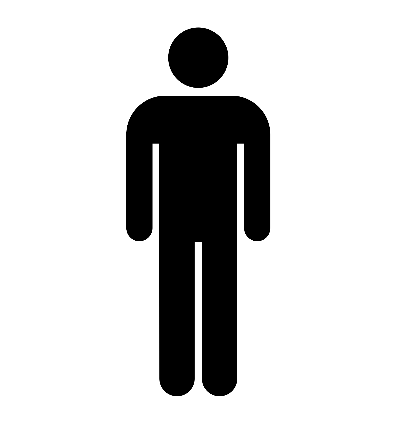
Gameplay consists of forward/backward/left/right movement by the player along with being able to look around, like standard FPS controls.

The user also has a single interact input button.

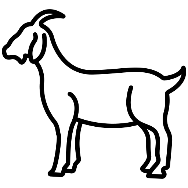
Overview of structure  
Player spawns into game scene, in front of them is 3 doors with buttons on stands, and to the side a host character.



Reset game and exit to menu button



Player chooses door by interacting with button.

Shape

Description automatically generated with low confidence

Host removes one door that DOES NOT have the prize and that the player has NOT selected.

Shape

Description automatically generated with low confidence

Player chooses to keep selection or change to other door, by pushing that door button

Shape

Description automatically generated with low confidence

Shape

Description automatically generated with low confidence

Host reveals winning door

Doors then reset in 10 sec. ready for the game to start again. Winning door randomly selected.

Note: Round 1, player listens to host instructions on how to play the game

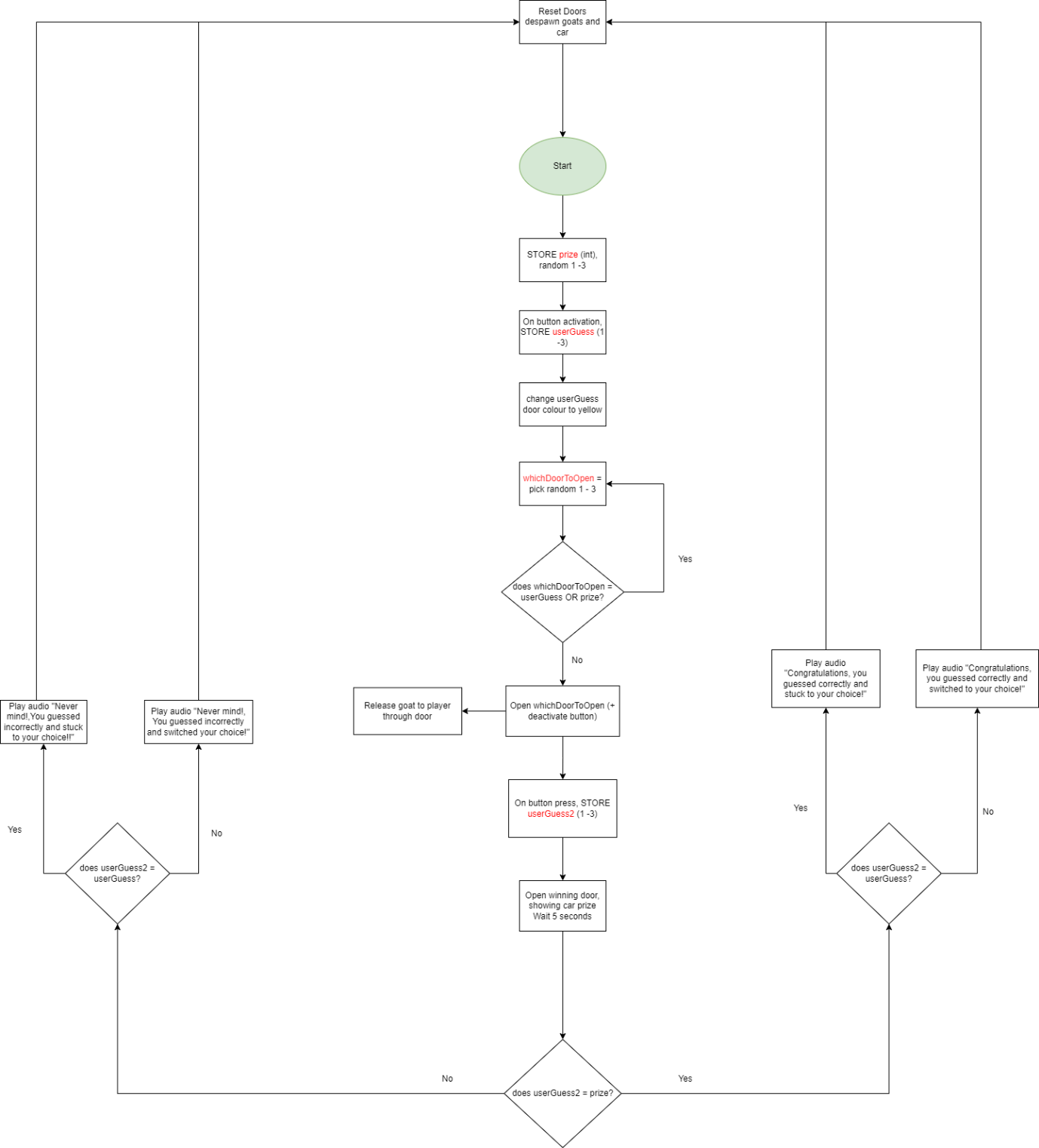
Note: Round 3, player listens to host explain the probability of the game and what the users best choice actually is.

Note: Game resets at round 5.

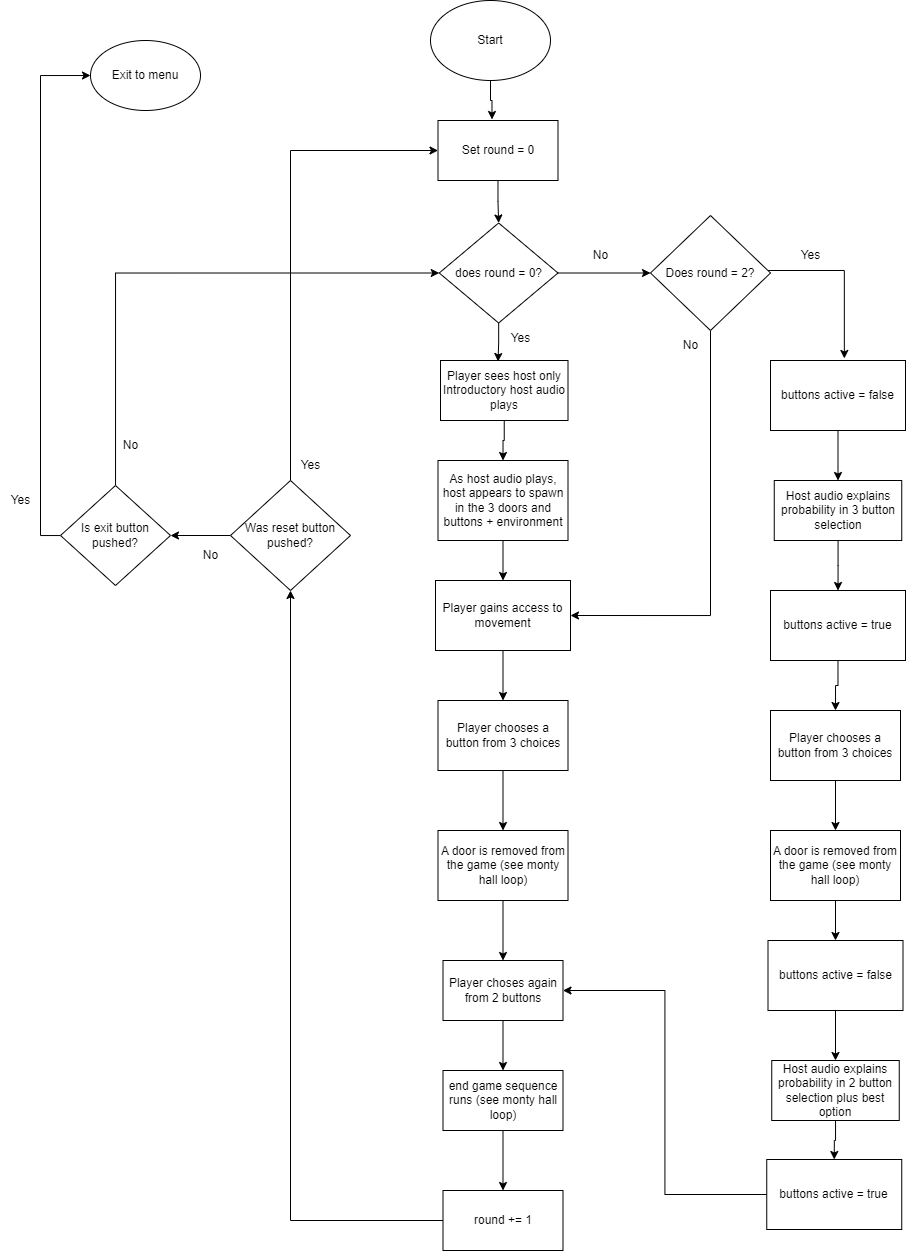
Gameplay Loops   
This section should showcase all gameplay loops, both for core gameplay, and any meta-loops too. It is vital the programming dept. understands what repeats, and what drives what so they can set up code structure to work with this. These can be taken from the GDD.

<MontyHall Loop>   
Add a few words to provide context for what it is, and of course the image of the loop.

This is a loop of how the game may run, regarding door selection



<Personal simulation loop>   
Loop for player gameshow simulation environment



<Scale Simulation Loop>   
Not added as yet for this task

# Gameplay Systems

This section of the document provides specifications for the systems that drive the game.

## Controls / Input

State the proposed control scheme for each platform. Include relavent diagrams and references materials to outline the control mechanisms required. It is optional to show controller mapping, it is more important you outline the connection between inputs and systems in the game.

If there are multiple areas that have a change in control scheme, outline the intended usage for each area. For example: Menu Navigation, vs Player controls. Remove / add subsections as applicable.

### Keyboard / Mouse

Describe

Used for movement (wasd) and look (mouse).

To interact with buttons, the user can push “E”, “F” or click the left mouse button

Escape key will bring up a menu to quit game or change input settings.

### 

### Controller

Describe

Left stick for translation, right stick to look.

Anyone of the main buttons (a,x,y,b) for interact.

The start will bring up a menu to quit game or change input settings.

### 

### Mobile / Touch

Bottom left for movement, bottom right of screen for look

Touch button on screen to interact if within the required distance.

A menu button will be placed in the top right corner of the screen to allow interactions to quit the application or change input settings.

## Game Mechanics

All gameplay mechanics are to be covered here. You may format information as you like, but ensure you cover the info for the bullet points listed under the examples below. Keep information concise – This is reference material, not justification, etc.

### <Name of Mechanic 1>

* Description of mechanic, and its purpose.
* Details of workings / rules of mechanic. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

### /// Repeat as necessary.

### <Movement >

* This will allow the player to move around in a 3D space. This is required to give a feeling of freedom in the game environment and allow players to walk up to things they wish to interact with.
* There is only one speed of movement as the playable environment is relatively small.
* Inputs need to allow for movement and looking in which ever direction the player desires
* The output is the players position on the map.

### <Interact >

* This will allow the player make choices for the game.
* Choices include selecting a button to push or to pat a goat.
* Inputs will simply be pushing the interact button with the player is near a button or animal object.
* The output depends on the button pushed: either selecting a door and changing its colour for player feedback on their choice, or to reset or exit the game at the end of the round.
* Output on a goat will result in the player patting the goat.
* “pat goat” interact is only suggested as a ‘feel good’ mechanic and doesn’t add to the development of the game.

/// End repeated section.

## Custom Game Systems

This covers systems that help drive and support gameplay that may not be a mechanic itself. For example, an inventory system may co-ordinate with mechanics such as picking up and dropping items (actions). A chat system may really be just interface systems required for the actual chat mechanics to be usable by the player. You may also have some systems that do things to your game state or level that are not a mechanic, or are simply purely technical in nature.

You MUST cover event handlers and UI systems here!

### <Host speech>

* The host is in the game to provide advice and learning to the player regarding the game and probability. The host also acts to provide atmosphere and possibly some humour for the player.
* There are no inputs with use of the gameshow host, but there are moments where gameplay pauses so the player will listen to what the host is saying. See personal simulation loop for more details.
* The output is in the form of audio from triggers in the game.

### /// Repeat as necessary. <Name of System X>

* Description of system, and its purpose.
* Details of workings / algorithms of system. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

/// End repeated section.

## 

## Physics <if applicable>

Very simply, identify how physics work in the game. If gameplay involves conscious collision of physical objects in a level, this is worth covering to ensure you understand how physics work in your game, even if that’s just out-of-the-box stuff from your game engine.

Keep this section concise. Use bullet points where possible, just ensure context is clear – Someone else should find this informative enough to know things.

The objects in the game: doors, buttons and goats will need to have collisions enabled as they are physical objects and walking through them may ruin the experience of the game.

## Behaviours / AI <if applicable>

AI is not applicable for this experience.

# Game Content

This section of the document covers content types for the project, and provides technical specifications on their usage.

## Game Environment

Provide a summary of the nature of the gameplay environments. Does everything take place in a fixed environment? Does the player move through multiple 3D levels? Provide information in a paragraph or two, and add supporting bullet-points if useful/necessary.

This game will run in a fixed size environment but with a feeling of space where to player can see a skybox but also no move too far out of the game area.

The rest of this section is about ensuring you are clear on the technical specs and usage of your content types.

/// Repeat as necessary – Add any supporting sub-headings or infor required, such as diagrams, etc

## <Game Content Type X>

* Concise description of content type (just to clarify) – 1 line at most.
* Location in project (actual file location / folder structure, or file(s) the content is stored in)
* Application details (tech specs / format requirements / required processes)

### List of <content type>

* <Item 1>
* <Item 2>
* etc

/// End repeated section.

# Coding Standards

Outline the coding conventions followed during the development of the project.  
You may link to existing coding standard documentation

## Coding Standards - Details

Actually bullet-list some of your programming standards. Showing images and providing examples to help clarify is good.

* What are your criteria for when to comment?
* How do you nest brackets for sections of code?

## Naming Conventions

You should have rules on how things are named to ensure consistency and ease of understanding of intent with naming.

* How do you name variables and functions?
* List any prefixes or suffixes you use for naming.

# Technical Goals & Risks

Bullet-point any Technical Goals that can be identified for the development of this project. A goal statement should identify how it is measured for success.

Example (Goals):

* Maintaining 60 FPS on min spec webgl build with desired graphics / shaders features
* Creating configurable inventory and collection system
* Saving the game at checkpoints

## Technical Goals:

* Bullet point technical goals

## Technical Risks:

* Bullet point technical risks (skill gaps, or potential over scoped features, areas of project needing additional research)

## Risk Avoidance:

For each goal / risk outline potential approaches that can be taken to minimize the risk. Area there features that could be cut / redesigned?

# Appendix A – Technical choice justifications

This appendix serves as a place to justify some of the development environment choices you made. THIS IS IMPORTANT FOR ASSESSMENT. Following the red text under each heading, you can see what information you need to show here – This is information you will find by performing some basic research on programming languages, and game engines.

## Choice of development engine:

* Justify the choice of the engine you are using for your project. This must provide logic / reasoning as to why you would use this engine.
* Have more than one reason. Reasons can be very basic and practical based on your circumstance, but at least one must connect with qualities of the development engine – Something to show you could compare it to others.

## Choice of scripting language:

* Justify the choice of the scripting language you are using for your project (c#). This must provide logic / reasoning as to why you would use this language.
* Have more than one reason. Reasons can be very basic and practical based on your circumstance (like what engine you are working with), but at least one must connect with qualities and capabilities of the scripting language – Something to show you could compare it to others.

## Choice of third-party libraries and content: (if required)

* Justify the choice of any 3rd party libraries and content used in the project. This must provide logic / reasoning as to why these were good choices to make for your project.