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

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.

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 |
| 0.0.4 | Harman | 4/4/23 | Updated flowchart details regarding variables. Added technical details to coding standards and technical choice justifications |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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:

* **Type**: Mention the updates or removal of functionality of the game
* **Scope:** Should be mentioned in terms of assessment implementation
  + Script
  + Level
  + TDD or Scripting documentation update
* **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 |
| “Added documentation” (for document updates only) |

# Game Overview

## Description

## 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

This is a first person game to increase engagement and sense of scale.

## Target Platforms

This project will be deployed to the following platforms:

* Browser

### Minimum <Platform> Specs

### 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:

* <PlayerSimulation mode (see below)>Interactable Buttons (x5):

x3 for doors

x1 for reset

x1 for peeking at answer before progressing (to be revised)

* <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.
* Prizes that spawn

# Game Flow & Structure

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

Game Modes & Handling

* Menu screen (GUI)
* Run personal simulation + exit condition to Menu
* Run scale simulation + exit condition to Menu

Game Mode - <MenuScreen> 

Description   
  
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 

Get the user to select required game state.

Objective Tracking 

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 

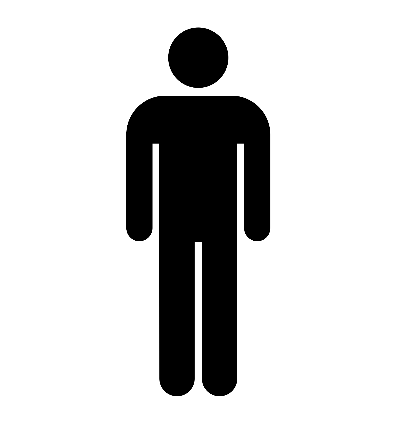
1. Save the number of simulations
2. Save the stay or swap choice
3. Run the simulation and instantiate objects for every winning round?

Mission / Level Structure

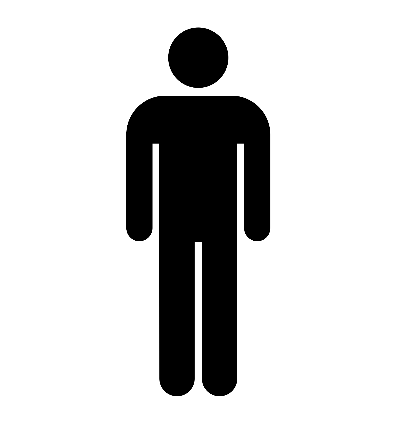
Gameplay consists of forward/backward/left/right movement by the player along with being able to look around, like standard FPS controls.

The user also has a single interact input button.

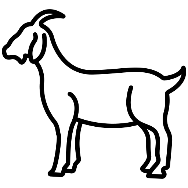
Overview of structure  
Player spawns into game scene, in front of them is 3 doors with buttons on stands, and to the side a host character.



Reset game and exit to menu button



Player chooses door by interacting with button.

Shape

Description automatically generated with low confidence

Host opens 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.

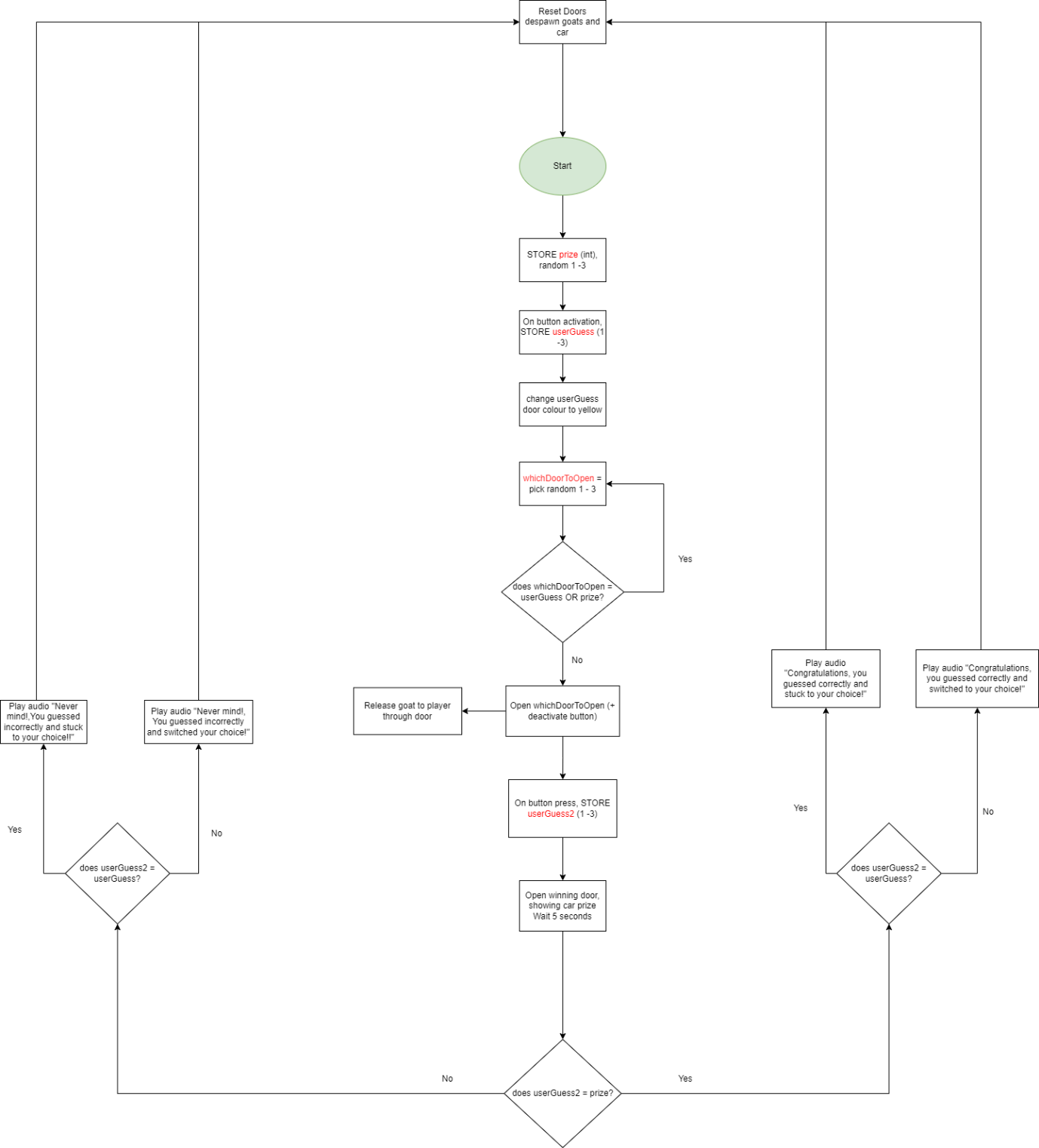
Note: Round 1, player listens to host instructions on how to play the game

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

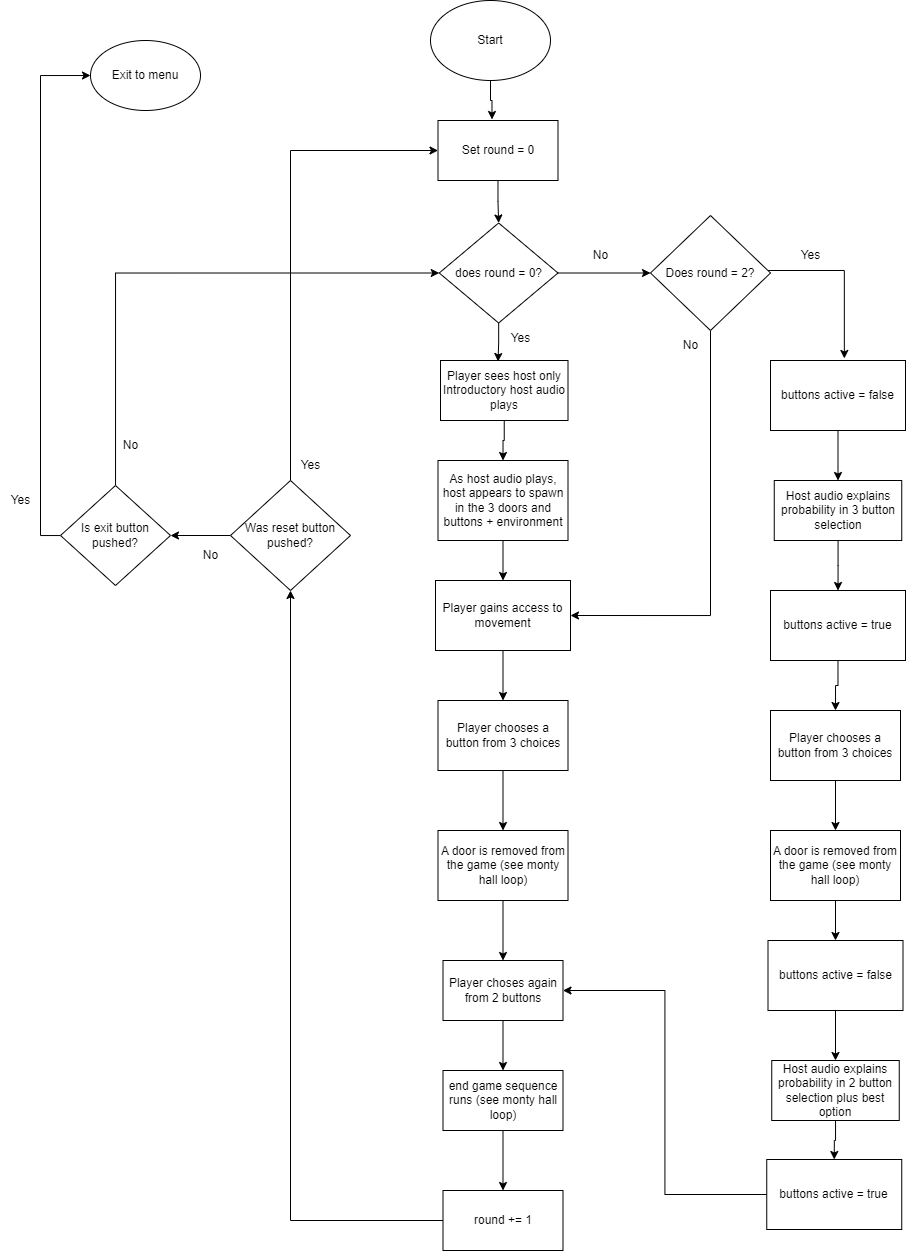
Note: Game resets at round 5.

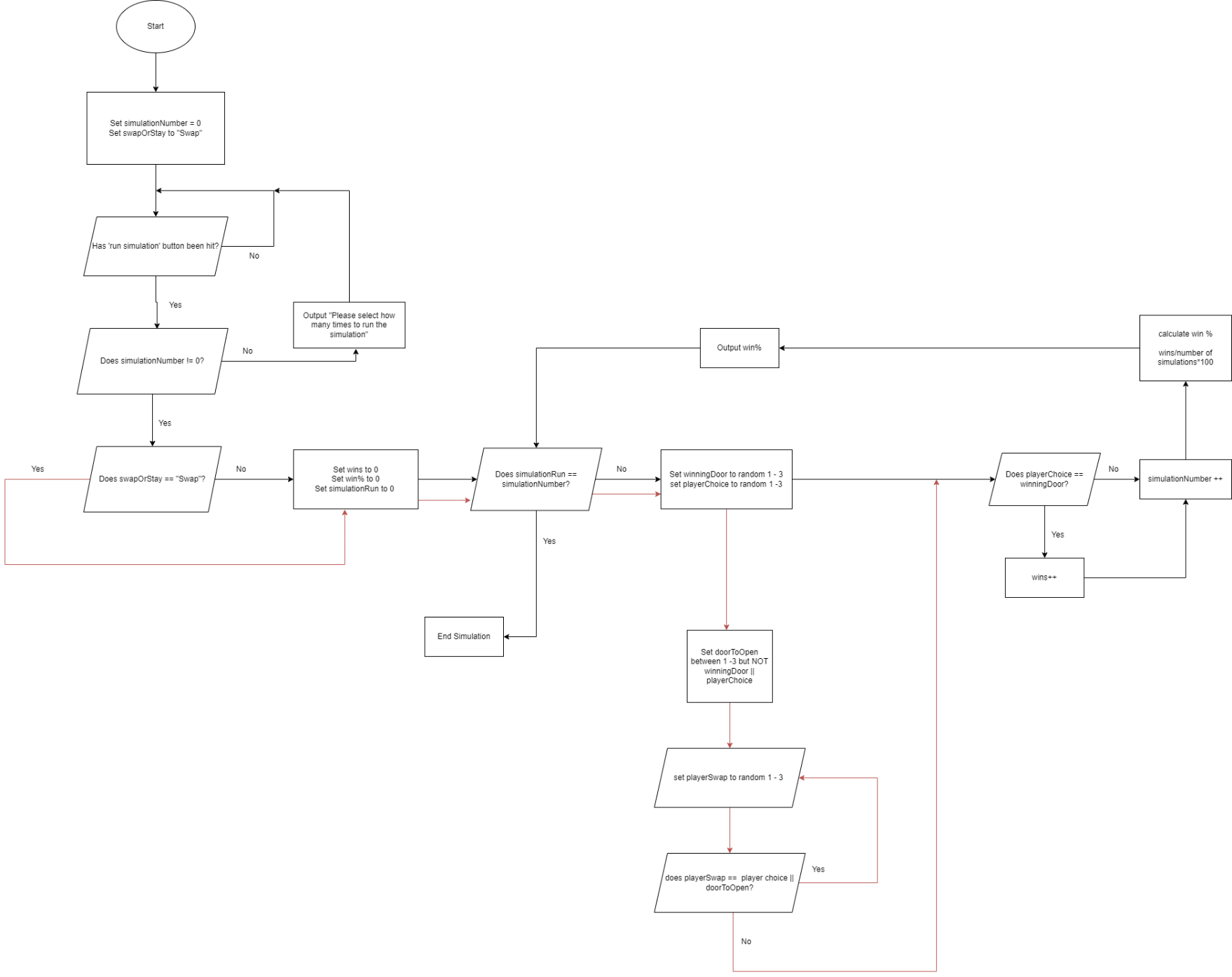
Gameplay Loops   
<MontyHall Loop>

This is a loop of how the game may run, regarding door selection and end of round audio.



<Personal simulation loop>   
Loop for player gameshow simulation environment. Changes in audio based on round number.



<Scale Simulation Loop>

# Gameplay Systems

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

## Controls / Input

### 

### Keyboard / Mouse

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

To interact with buttons, the user will simply walk up to the button

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

### 

### Controller

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

### <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.
* In game, Inputs will simply be the player walking in close proximity to the buttons
* For menu systems, players can use the mouse and left click to select a desired option

## Custom Game Systems

### <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 and timers in the game.

### <Timers>

* Used for ques with host speech and to keep the pace of the game reasonable with opening doors and button selections.
* Different audio will play depending on the round number and buttons selected.

<Round numbers>

* These will be tracked so the player can experience the game and be informed after a number of attempts of the probabilities in the experience.
* See <personal simulation loop> for rules.
* For rounds to be counted, the game will also need to be reset once complete.
* The outputs of the round counter will alter the host audio that plays.

## Physics <if applicable>

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

Collisions with buttons will also trigger material colour changes and updates in variables of player selections. Box colliders will be used to detect when the player is in close proximity to a button.

## 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

This game will run in a fixed size environment but with a feeling of space where to player can see a skybox but also not move out of the game area.

The game will have a small amount of assets and feature a host who will talk the player through the experience, trigged by round numbers and game timers.

## <Monty Hall Game content>

* Content will be a mixture of simple shapes for buttons and imported assets from the Unity asset store, being the prizes and doors.
* C:\Monty\_Hall\_Assessment or see GitHub repository.
* Will need to be packaged for use as a web-based experience, and settings/preferences in Unity will need to be correct to achieve this.

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

### List of Assets

* <Button 1>
* <Button 2>
* <Button 3>
* Host model
* Doors with animation (x3)
* Reset button
* Exit button
* Floor
* Skybox

# Coding Standards

## Coding Standards - Details

Some coding practices followed can be found here:

<https://avangarde-software.com/unity-coding-guidelines-basic-best-practices/>

In gereral, some main standards include:

* Classes and Methods in PascalCase
* Fields and variables in camelCase
* Braces on their own lines, and line up with corresponding open/close
* Comments should be added before a sequence of code that performs a specific function, or as deemed necessary for clarification.
* Add access level modifiers, even though defaulted to private.

Example below:

//save and load round numbers using PlayerPrefs

public void SaveRound()

{

PlayerPrefs.SetInt("Round", roundNumber = roundNumber++);

}

public void LoadRound()

{

roundNumber = PlayerPrefs.GetInt("Round");

}

## Naming Conventions

Variables and functions should be named appropriately for their task or provide information of the significance of the data. These should also be kept to a max length of 15 characters/3 words where possible.

e.g.

public int winningDoor;

public int playerChoice;

public int swapOrStay;

Due to the small scale of the project and assets needed, prefixes and suffixes are not defined.

Naming of doors and buttons should align however, for example, having “button1” and “door1” relating to each other for actions.

Technical Goals & Risks

## Technical Goals:

* Creating rounds to be counted: different actions based on round number.
* Changing material colours on collision: visible change of colour and updated variables.
* Programming the simulation to work correctly: correct doors opening and prizes spawning.

## Technical Risks:

* Many different types of hardware run web-based experiences, so the game may not work on some low end devices.

Risk Avoidance:

* Have the game run on either low resolution or have the fidelity scalable based on hardware.
* Version control updated anytime significant changes are made to the hardware so it can be reverted.

# Appendix A – Technical choice justifications

## Choice of development engine:

* Unity engine used for the development of the experience.

This engine was used as the body of knowledge for programming and development is the greatest by the developer, work colleagues and assessors.

* Unity provides a large amount of flexibility for game development and the visual experience target isn’t high end, negating the need for more technical engines.
* Unity is free to use and easy to find answers to technical problems. This is important for skill development and when development budget is zero.

## Choice of scripting language:

* Unity works natively with C# and has a number of libraries programmed in this language.
* C# is easier to learn and develop skills with compared to some other languages, such as C++, which can require extra programming for memory management.
* While languages like python are easy to learn and generally have a smaller syntax learning curve, it is not natively supported by Unity. There is a plug-in that can be used to program in Unity using python, but it has been reported that there is little support compared to C#.

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

* No 3rd party libraries
* Some assets used from the Unity asset store to save time to allow extra focus on the development of the experience.