# **SCRIPT DESIGN – Monty Hall Probability Game**

Sion Harman

Contents

[SCRIPT DESIGN - <Monty Hall Probability Game> 1](#_Toc69075429)

[Document overview 4](#_Toc69075430)

[Project Script Requirements 4](#_Toc69075431)

[Description 4](#_Toc69075432)

[Feature & Mechanics List 4](#_Toc69075433)

[Key Scripts to Design 4](#_Toc69075434)

[Script Design 5](#_Toc69075435)

[Script 1 - <MontyHallGame / gameflow and timing> 5](#_Toc69075436)

[Required Functionality & Outcomes 5](#_Toc69075437)

[Pseduocode 5](#_Toc69075438)

[Flowchart 5](#_Toc69075439)

[Script 1 Plan feedback 5](#_Toc69075440)

[Pseudocode feedback notes 5](#_Toc69075441)

[Flowchart feedback notes 5](#_Toc69075442)

[Script 1 Revised Plans 5](#_Toc69075443)

[Final pseudocode 5](#_Toc69075444)

[Final flowchart 6](#_Toc69075445)

[Script 2 - <Button1/change material colour/disable buttons> 6](#_Toc69075446)

[Required Functionality & Outcomes 6](#_Toc69075447)

[Pseduocode 6](#_Toc69075448)

[Flowchart 6](#_Toc69075449)

[Script 2 Plan feedback 6](#_Toc69075450)

[Pseudocode feedback notes 6](#_Toc69075451)

[Flowchart feedback notes 6](#_Toc69075452)

[Script 2 Revised Plans 6](#_Toc69075453)

[Final pseudocode 6](#_Toc69075454)

[Final flowchart 7](#_Toc69075455)

[Script Implementation & Iteration 7](#_Toc69075456)

[Script 1 – MontyHallGame/flow of game mechanice 7](#_Toc69075457)

[Script(s) generated 7](#_Toc69075458)

[Functionality review 7](#_Toc69075459)

[Implementation feedback notes 7](#_Toc69075460)

[Response to feedback notes 7](#_Toc69075461)

[Script 2 – Button1/change colour of button on contact, depenent on timers 8](#_Toc69075462)

[Script(s) generated 8](#_Toc69075463)

[Functionality review 8](#_Toc69075464)

[Implementation feedback notes 8](#_Toc69075465)

[Response to feedback notes 8](#_Toc69075466)

Document overview

The purpose of this document is to show the planning, design, implementation of at least two scripts in a single project. The document is broken up into phases to go through in order.

Project Script Requirements

### Description

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 in an interactive way.

The game will be like a game show, in First Person. A completed experience would include a host that is interacting/talking with the player and explaining the probability of the scenario.

### Feature & Mechanics List

* **Moving around/ Movement/Look** - <The player can move and have free look controls, using the input method appropriate for the platform. Used 1st person controller template for this>
* **Button Select / Change colour** - < Once a player collides with a button, it will change colour to provide feedback that a selection has been made>
* **Button Select / Trigger door to open** - < Once a player has selected a door via a button, the game has one door being opened>
* **Button Select / Reset –** the remaining buttons associated with the remaining closed doors will need to remain selectable, while the button of the opened door will need to be inactive
* **Button deactivate/deactivate button** – needed so a player can’t select multiple buttons
* **Spawn prizes / instantiation of prizes** – needed to show what was behind the door.
* **Host advice / trigger audio** – needed so the host can provide timely information to the player regarding the game

### Key Scripts to Design

* **Script 1** - <Game script that creates and updates the variables needed for the game, including randomly choosing the winning door, tracking the player guess and assigning a door to reveal to the player. This will also control the timing and flow of the game, along with keeping track of round numbers using PlayerPrefs>
* **Script 2** - <Script that changes the colour of the button to green when a player collides with the button>

## Script Design

### Script 1 – MontyHallGame / controlling game variables.

#### Required Functionality & Outcomes

The script will need to handle the main events of the game, including tracking variables for the scenario and timer implementation. These variables will need to be public so they can be called on by other scripts to create actions.

Variables:

* Winning door (random int 1-3 relating to the 3 doors)
* Player Choice (int: updated on collision with button)
* Door to open (int: chosen by not revealing the players choice, or the winning door)
* Timer (float: to provide time for the host to deliver information to the player after they have made their selection, also so the door isn’t revealed instantaneously)

#### Pseudocode

Declare Variables

Winning door: select random number between 1 and 3

Set player choice to zero.

**If player has made a choice**:

set *door to open* number between 1 and 3, but can’t equal the players choice, or winning door number.

Deactivate all buttons

Start timer

**When timer == 0**

Open door animation plays

Spawn goat at open door

Change player choice button to yellow

Activate remaining door buttons

Record player choice in Swap or Stay variable

**When swap or Stay Variable != 0**

Open winning door

Reveal Sports Car at winning door

If swapOrStay equals playerChoice

If swapOrStay equals winningDoor

Play “you won and stuck to your guns” audio

Else play “You lost but stuck to your choice” audio

If swapOrStay does not equal playerChoice

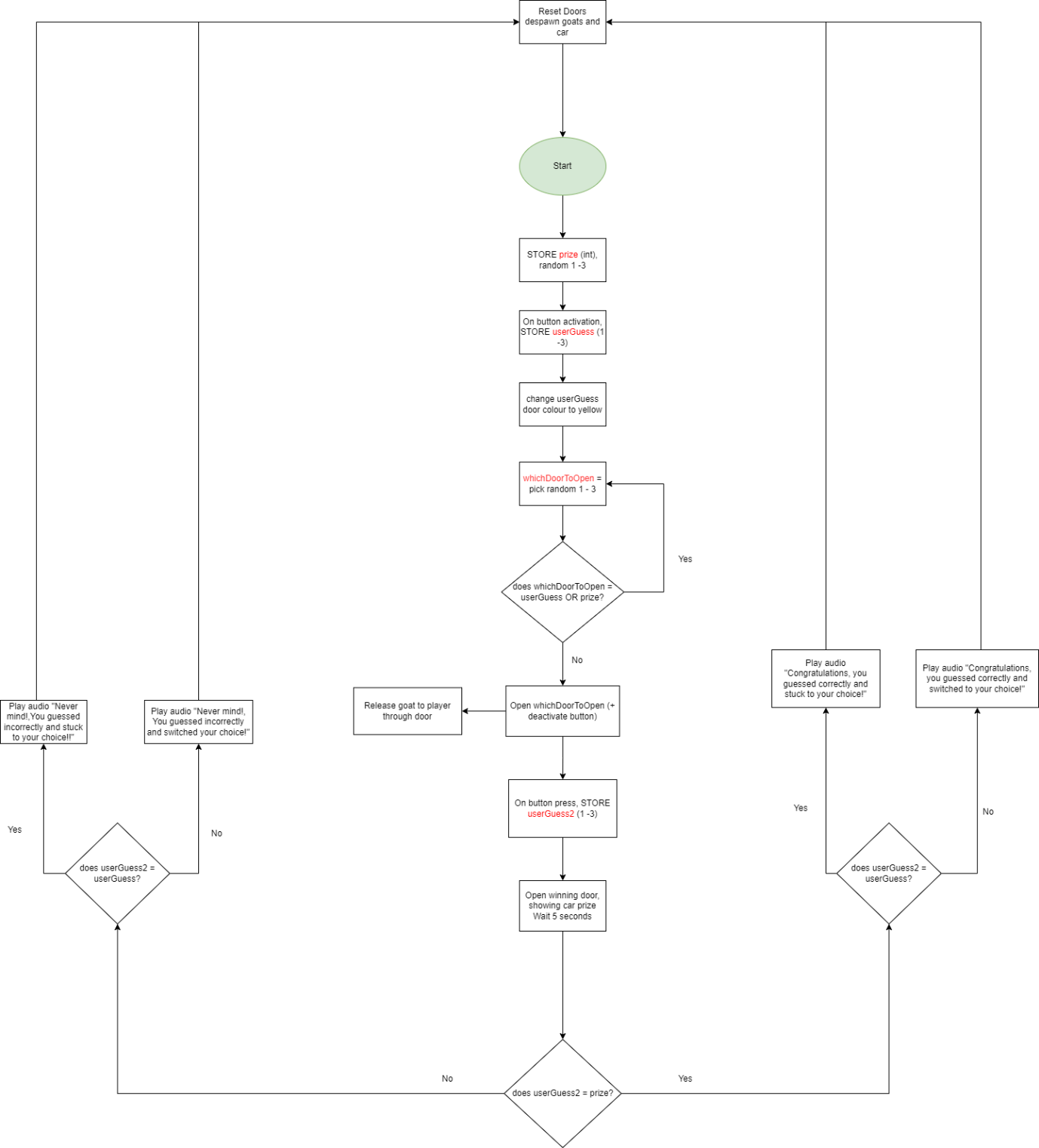
If swapOrStay equals winningDoor

Play “You won by switching your choice!” audio

Else play “You lost by switching your choice!” audio.

Game resets

#### Flowchart



### 

### Script 1 Plan feedback

#### Pseudocode feedback notes

* Nil

#### Flowchart feedback notes

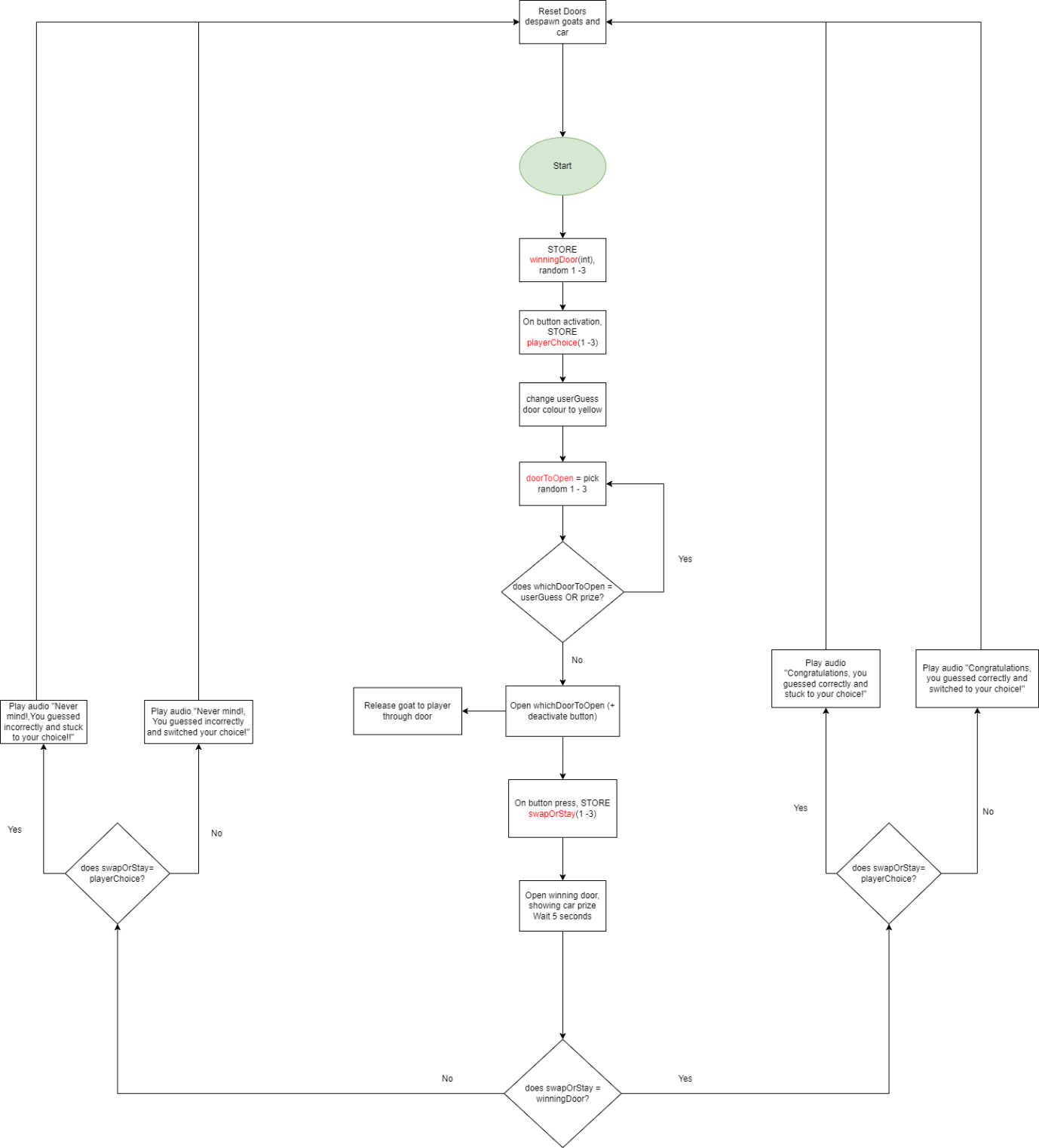
* Update variable names to better reflect data being stored

### Script 1 Revised Plans

#### Final pseudocode

As above

#### Final flowchart



### Script 2 - <Button1/change colour of button/disable buttons>

\*\*Please note scripting for this is split over the 3 button Scripts (Button1, Button2 and Button3) and MontyHallGame Script (for timing of buttons for colours).

#### Required Functionality & Outcomes

The script will be required to change the colour of a button that has been collided with by the player. There will need to be 2 colours, one representing the players initial choice, and another their final choice.

* Will need an on-collision method to check if the button has collided with the player
* Will need to change the material colour to the first colour (yellow)
* Check if a certain amount of time has expired
* Check for another collision to turn the button choice to the final colour (green)
* Once final choice made, disable the other button/s

#### Pseudocode

Check if the button has collided with the player

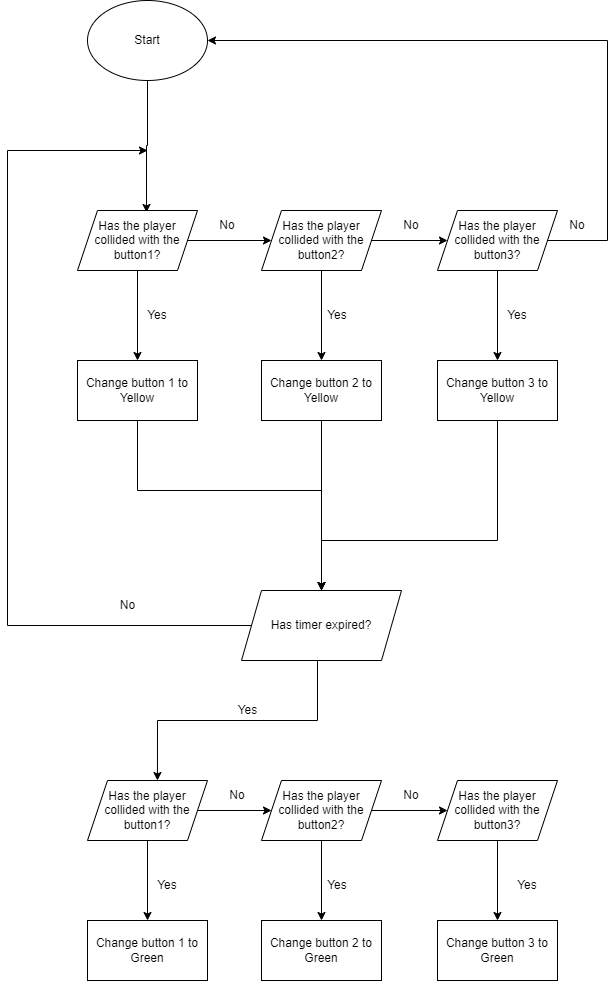
If collided, change the material colour to yellow/orange

Check to see if timer is <= zero

If collided, change the material colour to Green

Disable other buttons

#### Flowchart



### 

### Script 2 Plan feedback

#### Pseudocode feedback notes

* Nil

#### Flowchart feedback notes

* Nil

## Script Implementation & Iteration

### Script 1 - <Monty Hall Game script/function>

#### Script(s) generated

public class MontyHallGame : MonoBehaviour

{

public int winningDoor;

public int playerChoice;

public int swapOrStay;

public int doorToOpen;

public int expiredTimer = 1;

public int carSpawn = 0;

public int roundNumber = 0;

//public Button Button1;

//public Button Button2;

//public Button Button3;

public DestroyDoor openDoor, spawnCar, doorAnimate;

public GameObject NewCar;

private bool gameRestarting;

private bool carSpawned = false;

public float sceneEndTimer = 5;

public float timer = 400;

public float round2Timer = 2;

public float round3Timer = 5;

// Start is called before the first frame update

void Start()

{

//pick a winning door

winningDoor =

Random.Range(1, 4);

roundNumber = PlayerPrefs.GetInt("Round") + 1;

//Reset game after round 5

if (roundNumber > 4)

{

PlayerPrefs.SetInt("Round", roundNumber = 1);

}

}

//save and load round numbers using PlayerPrefs

public void SaveRound()

{

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

}

public void LoadRound()

{

roundNumber = PlayerPrefs.GetInt("Round");

}

// Update is called once per frame

void Update()

{

//pick a door to open, that is not the player choice or the winning door

while(doorToOpen == winningDoor ||doorToOpen == playerChoice)

{

doorToOpen = Random.Range(1, 4);

}

//Once the player has made a choice, start the first timer

if (playerChoice != 0)

{

timer -= Time.deltaTime;

if (timer < 0)

{

timer = 0;

expiredTimer= 0;

Debug.Log("You chose door " + playerChoice);

}

}

if(expiredTimer == 0)

{

//round 2 timer is added to provide instructions and a delay before remaining buttons are reactivated

if (round2Timer > 0)

{

round2Timer -= Time.deltaTime;

if (round2Timer < 0)

{

round2Timer = 0;

}

}

if (round2Timer == 0)

{

if (doorToOpen == 1)

{

//addB2Collider.AddComponent<SphereCollider>();

}

}

}

//add a restart to the scene once timer reaches zero or below and add one to the round number

if (sceneEndTimer > 0 && round3Timer == 0)

{

sceneEndTimer -= Time.deltaTime;

if (sceneEndTimer <= 0)

{

//SceneManager.LoadScene("Playground");

SaveRound();

Application.LoadLevel("Playground");

LoadRound();

Debug.Log("Round number is " + roundNumber);

//roundNumber = PlayerPrefs.GetInt("Round", roundNumber++);

}

}

//Once the player has made their second choice, start a timer to reveal the winning door and prize

if (swapOrStay != 0)

{

round3Timer -= Time.deltaTime;

Debug.Log("Your final choice is door " + swapOrStay);

if (round3Timer < 0)

{

round3Timer = 0;

while (winningDoor == 1 && carSpawned == false)

{

//spawn in the car and open the door

Instantiate(NewCar, spawnCar.spawnPoint1.transform);

doorAnimate.doorDestroy1.GetComponent<Animation>().Play();

carSpawned = true;

}

while (winningDoor == 2 && carSpawned == false)

{

Instantiate(NewCar, spawnCar.spawnPoint2.transform);

doorAnimate.doorDestroy2.GetComponent<Animation>().Play();

carSpawned = true;

}

while (winningDoor == 3 && carSpawned == false)

{

Instantiate(NewCar, spawnCar.spawnPoint3.transform);

doorAnimate.doorDestroy3.GetComponent<Animation>().Play();

carSpawned = true;

}

//Check winning conditions for the probability game and play relevent audio (here as a debug log)

if(swapOrStay == winningDoor)

{

if (playerChoice == swapOrStay)

{

Debug.Log("You chose correctly and stuck to your choice!");

}

else

{

Debug.Log("You chose correctly by changing your mind!");

}

}

else

{

if (playerChoice == swapOrStay)

{

Debug.Log("You didn't win this time and you stuck with your decision");

}

else

{

Debug.Log("You didn't win this time and you changed your choice!");

}

}

}

}

}

}

#### 

#### Functionality review

30/3/23

Script functions as desired in the minigame, but there are some errors with the output as shown in the debug log where the round number output from PlayerPrefs is a round behind.

1/4/23

Rounds now show correctly.

#### Implementation feedback notes

* Door destroying happens too fast
* No prizes spawning making it unclear
* Winning conditions on game end unclear in game
* No sound added
* Some unused variables declared

#### Response to feedback notes

* 24/3/23 Change door destroy to animate the door for a period of 1 second. Added timer to control round times
* 24/3/23 Added sheep instantiation to opened door.
* 24/3/23 added clearer winning conditions, including instantiation of car prize.
* 4/4/23 added sound to the script, along with audio control class.
* 5/4/23 removed unused variables and classes.

#### FINAL SCRIPTS including AudioScript

**Monty Hall Game script:**

using JetBrains.Annotations;

using System.Collections;

using System.Collections.Generic;

using System.Runtime.CompilerServices;

using Unity.VisualScripting;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class MontyHallGame : MonoBehaviour

{

public int winningDoor;

public int playerChoice;

public int swapOrStay;

public int doorToOpen;

public int expiredTimer = 1;

public int carSpawn = 0;

public int roundNumber = 0;

public DestroyDoor openDoor, spawnCar, doorAnimate;

public GameObject NewCar;

public AudioScript sound;

public Audio2Script sound2;

private bool carSpawned = false;

public float sceneEndTimer = 6;

public float timer = 11;

public float round2Timer = 6;

public float round3Timer = 6;

// Start is called before the first frame update

void Start()

{

//pick a winning door

winningDoor =

Random.Range(1, 4);

roundNumber = PlayerPrefs.GetInt("Round") + 1;

//Reset game after round 5

if (roundNumber > 4)

{

PlayerPrefs.SetInt("Round", roundNumber = 1);

}

sound.PlayAudio();

sound.ExplainInitial();

}

//save and load round numbers using PlayerPrefs

public void SaveRound()

{

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

}

public void LoadRound()

{

roundNumber = PlayerPrefs.GetInt("Round");

}

// Update is called once per frame

void Update()

{

//pick a door to open, that is not the player choice or the winning door

while(doorToOpen == winningDoor ||doorToOpen == playerChoice)

{

doorToOpen = Random.Range(1, 4);

}

//Once the player has made a choice, start the first timer

if (playerChoice != 0)

{

timer -= Time.deltaTime;

if (roundNumber != 3)

{

sound.PlayAudio2();

}

if (roundNumber== 3)

{

sound.ExplainSecondChoice();

}

if (timer < 0)

{

timer = 0;

expiredTimer= 0;

Debug.Log("You chose door " + playerChoice);

//sound.PlayAudio2();

}

}

if(expiredTimer == 0)

{

//round 2 timer is added to provide instructions and a delay before remaining buttons are reactivated

if (round2Timer > 0)

{

round2Timer -= Time.deltaTime;

if (round2Timer < 0)

{

round2Timer = 0;

}

}

}

//add a restart to the scene once timer reaches zero or below and add one to the round number

if (sceneEndTimer > 0 && round3Timer == 0)

{

sceneEndTimer -= Time.deltaTime;

if (sceneEndTimer <= 0)

{

//SceneManager.LoadScene("Playground");

SaveRound();

Application.LoadLevel("Playground");

LoadRound();

Debug.Log("Round number is " + roundNumber);

//roundNumber = PlayerPrefs.GetInt("Round", roundNumber++);

}

}

//Once the player has made their second choice, start a timer to reveal the winning door and prize

if (swapOrStay != 0)

{

round3Timer -= Time.deltaTime;

Debug.Log("Your final choice is door " + swapOrStay);

if (round3Timer < 0)

{

round3Timer = 0;

while (winningDoor == 1 && carSpawned == false)

{

//spawn in the car and open the door

Instantiate(NewCar, spawnCar.spawnPoint1.transform);

doorAnimate.doorDestroy1.GetComponent<Animation>().Play();

carSpawned = true;

}

while (winningDoor == 2 && carSpawned == false)

{

Instantiate(NewCar, spawnCar.spawnPoint2.transform);

doorAnimate.doorDestroy2.GetComponent<Animation>().Play();

carSpawned = true;

}

while (winningDoor == 3 && carSpawned == false)

{

Instantiate(NewCar, spawnCar.spawnPoint3.transform);

doorAnimate.doorDestroy3.GetComponent<Animation>().Play();

carSpawned = true;

}

//Check winning conditions for the probability game and play relevent audio (here as a debug log)

if(swapOrStay == winningDoor)

{

if (playerChoice == swapOrStay)

{

Debug.Log("You chose correctly and stuck to your choice!");

sound.WinStay();

}

else

{

Debug.Log("You chose correctly by changing your mind!");

sound.WinSwap();

}

}

else

{

if (playerChoice == swapOrStay)

{

Debug.Log("You didn't win this time and you stuck with your decision");

sound.LoseStay();

}

else

{

Debug.Log("You didn't win this time and you changed your choice!");

sound.LoseSwap();

}

}

}

}

}

}

**Referenced AudioScript:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Audio;

public class AudioScript : MonoBehaviour

{

public MontyHallGame round;

public AudioSource audioSource;

public AudioClip clip;

public AudioClip clip2;

public AudioClip winSwap;

public AudioClip winStay;

public AudioClip loseStay;

public AudioClip loseSwap;

public AudioClip explainRound1;

public AudioClip explainRound2;

public AudioClip intro;

public const string audioName = "Introduction Audio.mp3";

private bool switchedOn = true;

private bool endAudio = true;

private bool explain2 = true;

//Play intro audio only on first round

public void PlayAudio()

{

if (round.roundNumber == 1)

{

GetComponent<AudioSource>().clip = clip;

GetComponent<AudioSource>().Play();

}

}

//Play round 2 audio, every round

public void PlayAudio2()

{

if (switchedOn == true)

{

GetComponent<AudioSource>().clip = clip2;

GetComponent<AudioSource>().Play();

switchedOn = false;

}

}

//code for audio to play based on winning conditions

public void WinSwap()

{

if(endAudio== true)

{

GetComponent<AudioSource>().clip = winSwap;

GetComponent<AudioSource>().Play();

endAudio = false;

}

}

public void WinStay()

{

if (endAudio == true)

{

GetComponent<AudioSource>().clip = winStay;

GetComponent<AudioSource>().Play();

endAudio = false;

}

}

public void LoseStay()

{

if (endAudio == true)

{

GetComponent<AudioSource>().clip = loseStay;

GetComponent<AudioSource>().Play();

endAudio = false;

}

}

public void LoseSwap()

{

if (endAudio == true)

{

GetComponent<AudioSource>().clip = loseSwap;

GetComponent<AudioSource>().Play();

endAudio = false;

}

}

//code for round 3 audio explaining the probabilities

public void ExplainInitial()

{

if (round.roundNumber == 3)

{

GetComponent<AudioSource>().clip = explainRound1;

GetComponent<AudioSource>().Play();

}

}

public void ExplainSecondChoice()

{

if (explain2 == true)

if(round.roundNumber == 3)

{

GetComponent<AudioSource>().clip = explainRound2;

GetComponent<AudioSource>().Play();

}

explain2 = false;

}

}

### Script 2 - <Button1>

#### Script(s) generated

This script also interacts with the “MontyHallGame” script as

using JetBrains.Annotations;

using System.Collections;

using System.Collections.Generic;

using System.Diagnostics.Contracts;

using UnityEngine;

public class Button1 : MonoBehaviour

{

public MontyHallGame plyrCh, timerCheck,secondChoice,openedDoor;

public Button2 disableButton2;

public Button3 disableButton3;

void OnCollisionEnter(Collision collision)

{

Debug.Log(collision.gameObject.name);

if (collision.gameObject.name == "PlayerCapsule")

{

Debug.Log("Yellow");

if (timerCheck.round2Timer > 0)

{

GetComponent<Renderer>().material.color = Color.green;

plyrCh.playerChoice = 1;

}

else

{

secondChoice.swapOrStay = 1;

GetComponent<Renderer>().material.color = Color.green;

//stop the player from selecting another button once final choice is made

disableButton2.GetComponent<SphereCollider>().enabled = false;

disableButton3.GetComponent<SphereCollider>().enabled = false;

}

}

#### Functionality review

28/3/23

The script generally functions as desired.

A shortcoming is that players on their initial choice can change their mind. If they change their mind to another button, the original button remains Yellow, which may be confusing to the player, particularly if they accidentally ran into a different button and changed their choice.

#### Implementation feedback notes

* Button always green for first and second choice makes it unclear.

#### Response to feedback notes

* Changed the first-round button press to yellow and second round push to green to make the final and initial choices more obvious. This was done using timers to determine the material colour

This script also interacts with the “MontyHallGame” script as

using JetBrains.Annotations;

using System.Collections;

using System.Collections.Generic;

using System.Diagnostics.Contracts;

using UnityEngine;

public class Button1 : MonoBehaviour

{

public MontyHallGame plyrCh, timerCheck,secondChoice,openedDoor;

public Button2 disableButton2;

public Button3 disableButton3;

void OnCollisionEnter(Collision collision)

{

Debug.Log(collision.gameObject.name);

if (collision.gameObject.name == "PlayerCapsule")

{

Debug.Log("Yellow");

if (timerCheck.round2Timer > 0)

{

GetComponent<Renderer>().material.color = Color.yellow;

plyrCh.playerChoice = 1;

}

else

{

secondChoice.swapOrStay = 1;

GetComponent<Renderer>().material.color = Color.green;

//stop the player from selecting another button once final choice is made

disableButton2.GetComponent<SphereCollider>().enabled = false;

disableButton3.GetComponent<SphereCollider>().enabled = false;

}

}