DropBox-Link: <https://db.tt/nFNuT72N>  
GitHub-Link:

Coursework Part A

Game Development

“The Gauntlet”

Due date: 23.10.2016

**Student:**

Name: Sören Schreiber

Kingston ID: K1549360

Module leader: Dr. Vasils Argyriou(VA)

Lecturers: -Dr. Vasilis Argzriou (VA)

Marker: - Dr. Vasilis Argzriou (VA)

**Table of Contents**

[**Introduction** 3](#_Toc464763301)

[**Assignment** 3](#_Toc464763302)

[**“The Gauntlet”** 3](#_Toc464763303)

[**Game Design** 3](#_Toc464763304)

[**Game Controls** 4](#_Toc464763305)

[**Implemented tasks** 5](#_Toc464763306)

[**Source code** 5](#_Toc464763307)

[**Scripts** 5](#_Toc464763308)

[ Script::Ball 5](#_Toc464763309)

[ Script::Bridge 6](#_Toc464763310)

[ Script::cameraControl 7](#_Toc464763311)

[ Script::ChargePlayer 7](#_Toc464763312)

[ Script::GameController 8](#_Toc464763313)

[ Script::gameStart 9](#_Toc464763314)

[ Script::GateControl 9](#_Toc464763315)

[ Script::GeneratorControl 10](#_Toc464763316)

[ Script::RespawnScript 11](#_Toc464763317)

[ Script::SimonSays 11](#_Toc464763318)

[ Script::SSChoiceB 12](#_Toc464763319)

[ Script::SSChoiceF 13](#_Toc464763320)

[ Script::SSChoiceL 14](#_Toc464763321)

[ Script::SSChoiceR 14](#_Toc464763322)

[ Script::SSStart 15](#_Toc464763323)

[ Script::teleportation 16](#_Toc464763324)

[ Script::transporterControlScript 16](#_Toc464763325)

[**Results** 17](#_Toc464763326)

[**Annex** 19](#_Toc464763327)

[**Source-Codes** 19](#_Toc464763328)

[**Table of Code segments** 42](#_Toc464763329)

[**Table of Figures** 43](#_Toc464763330)

[**Table of Tables** 43](#_Toc464763331)

[**References** 44](#_Toc464763332)

# **Introduction**

The first assignment for the 3d games development module focuses on the tools and mechanics taught during the first week of lectures. The Goal is to create a 2d or 3d game for the PlayStation system.

## **Assignment**

The assignment is to create a 2 dimensional or 3 dimensional game for a PlayStation 4 or mobile system, focusing on the available features and interaction mechanisms provided by the chosen platform.

The Games has to implement at least 1 of the following tasks:

1. Background. Include a (moving) background
2. Machinery. Put some interesting objects into your scene. To attract good marks, use non-trivial 2D/3D modelling, texturing or provide it with some kind of animated behaviour.
3. Vehicles/Moving Objects. Conceptually close to the machinery, but vehicles should be able to move around.
4. AI. If you’d like to try your hand in artificial intelligence, you are welcome! This is an open-ended topic.
5. Collision detection. You will have to introduce some kind of scenery to collide with. Collision detection should restrict players or NPCs motion based on contact conditions with other objects (e.g. walls).
6. Shooting. You will have to introduce some kind of target (enemy monsters?) and test the player’s accuracy. Extra bonus for moving targets!
7. Setting. This includes game title screen, menus, scoring, timing, status, hall of fame etc.
8. Anything else. You must obtain a go ahead from your Tutor if you have any other interesting idea.
9. Focus on a specific part and improve it as much as possible.

# **“The Gauntlet”**

## **Game Design**

“The Gauntlet” is a skill based single player puzzle game in which the player controls a ball throughout a level by solving puzzles and avoiding falling of the game platforms. The Goal of each level is to reach the other side and advance to the next level. Each level will present the player with new and increasingly difficult puzzles and game mechanics such as:

1. Jumping
2. Finding ways to activate bridges
3. Messing with gravity
4. Memory skills

The current version of the game has a total of 5 levels.

* Level 1: is a simple level designed to familiarize the player with the different basic controls (ball and camera movements) of the game.
* Level 2: is a more complicated area requiring skill to jump ridges, move the ball across narrow platforms as well as introducing power generators and energy fields to the player
* Level 3: is the most difficult level in the game. Challenging the player with a variety of different obstacles from the previous levels, as well as providing a new game mechanic which allows the player to change the direction of the gravity. Beating this level requires sensitive ball control as well as 3d thinking to make the basic ball controls move the ball where it is supposed to go.
* Level 4: is a simple level design opposing the player with a game of “simon says”. The player will be challenged to repeat given sequences (1-5) and has to complete all without being wrong once. If a wrong choice is made at any time the entire “simon says” will have to be reset and restarted from scratch. When the last sequence is entered correctly the final game will open allowing the player to reach the end game screen

## **Game Controls**

The game follows basic game controls using the 2 joysticks to control ball and camera movement as well as buttons for interactions and jumping.

For more details on the control scheme see the table below:

|  |  |  |
| --- | --- | --- |
| Control | Function | Description |
| LeftJoy-X | Ball(Left/Right) | The left joystick provides control for the player/ball.  The X-Axis will move the ball left and right in relation to the camera view |
| LeftJoy-Y | Ball(Forward/back) | The left joystick provides control for the player/ball.  The Y-Axis will move the ball forward and backward in relation to the camera view |
| RightJoy-X | Camera(Left/right) | The right joystick provides control over the camera.  The X-Axis will move the camera left and right around the ball. |
| RightJoy-Y | Camera(up/down) | The right joystick provides control over the camera.  The Y-Axis will move the camera up and down around the ball. |
| D-Pad-Y | Gravity(up/down) | The D-pad is only active while in level 3. It provides a manipulation tool for the gravity in the level.  The Y-Axis (up/down) will change the gravity in relation to the world Axis. |
| D-Pad-X | Gravity(Left/right) | The D-pad is only active while in level 3. It provides a manipulation tool for the gravity in the level.  The X-Axis (left/right) will change the gravity in relation to the camera view. |
|  | Interact | The interact button is used to interact with energy fields and generators as well as the input method for Level 4. |
|  | Transport | The transport button will activate the transporter to move the player form one level to the next. |
| R1 | Jump | This button will apply force to the ball making it jump. |
| L1 | Drop | Drops the charge from the player |
| Option | Exit | Close the application |

Table 1- Game Controls

## **Implemented tasks**

* Background: The skybox provides a visually nice background for the entire game
* UI: There are two different UI used in the game:
  + Start/End screen UI:

The start screen and the end game screen utilize a perspective Canvas element to tilt the UI elements giving it a perspective effect. It contains a Text and a button element.

* + Game UI:

The Game UI consists of two simple text elements providing information (such as level goals and interaction prompts) throughout the game.

* Machinery/Moving objects: The game has multiple animations for certain objects such as the transporters, bridges and generators. In addition there are also continuous particle emitters for the energy fields and a toggled particle system indication if the ball is charged with energy or not.
* Collison detection: besides standard walls, roof and floor collisions, the game entails multiple trigger detections for interactions between the player and word objects.
* Lighting/sound: throughout the game a background music will play until the 4th level of the game. When a “simon says” game is started the music will hold hand the sounds for the game will be played when prompted too. In addition there are 4 spotlights to indicate a visual reference to which colour is being played at that time.
* Setting: The game contains a start screen and an end game screen. For more information see point “UI”.

# **Source code**

The following section will describe all used scripts, the variables and the functions throughout the entire program.

For the full source codes see Annex section source-codes.

## **Scripts**

### Script::Ball

The “ball”-script provides the player with a control input to move the ball around the map. This includes basic movements, jumping as well as the gravity control in level 3.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Anchor | Public ParticleSystem | The particle system which visually indicates if the player currently carries a charge |
| GC | Public GameController | Contains the game controller script and allows access to its variables and functions |
| sideMovInput | Private float | Float variable reading the X-Axis input of the player |
| forwardMovInput | Private float | Float variable reading the Y-Axis input of the player |
| Rig | Private Rigidbody | RigidBody object of the ball. Used to apply torgue and thereby move the ball |
| Charged | Private bool | Boolean variable indication the current charge status |

Table 2 - Ball script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| start | -Input:  NONE  -Output:  NONE | Function is called on start of application.  Set charged value and reads rigidbody of ball |
| Update | -Input:  NONE  -Output:  NONE | Function is called each frame.  Reads any user input for the ball and moves the rigidbody accordingly.  Starts and stops the particle effect depending on charged value |
| Charge | -Input:  NONE  -Output:  NONE | Inverts the current charged status and saves it |
| readCharge | -Input:  NONE  -Output:  BOOL | Reads the current charged status and returns the value as a bool to the calling script |

Table 3 - Ball script functions

### Script::Bridge

This script enables a moving bridge to carry the ball along with its motion.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| NONE | NONE | NONE |

Table 4 - Bridge script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | When an object enters the trigger its parent will be set to this game object |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | When an object leaves the trigger, its parent will be set to NULL and rescaled to Vector3(4,4,4) |

Table 5 - Bridge script functions

### Script::cameraControl

The “cameraControl” provides the basic camera controls for the player as well as orientation for the camera to always be upright.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| camSideMov | Private float | Float variable reading the movement along the X-Axis from the player control |
| camUpMov | Private float | Float variable reading the movement along the Y-Axis from the player control |
| Player | Private GameObject | Player game object to be used for camera rotation |
| Anchor | Private GameObject | Anchor game object used to follow the ball and as a static point for the camera to look at |

Table 6 - Camera control script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| start | -Input:  NONE  -Output:  NONE | Function is called on the start of the game.  Find both player and anchor gameobject |
| Update | -Input:  NONE  -Output:  NONE | Function is called every frame.  Read user input to move camera and rotates the camera game object around the ball while making sure to follow the ball and to have the camera always upright |

Table 7 - Camera control script functions

### Script::ChargePlayer

Here the player will be charged with the energy needed to start a generator.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Player\_script | Public ball | Containing the ball script |
| Text | Public text | Containing a ui text object |

Table 8 - Charge player script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  NONE | Function is called each frame while a collider is inside the trigger.  When the interact button Is pressed and no charge is held the ball is charged |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Reset ui text to empty when the collider leaves the trigger |

Table 9 - Charge player script functions

### Script::GameController

The game controller is the main script of the game. It provides the current level, gives an option to close the application and changes the UI Layover and the camera culling mask.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Cam | Public cameraControl | Camera Control script |
| Player | Public ball | Ball script |
| StartCanvas | Public GameObject | GameObject containing the start/end UI |
| GameCanvas | Public GameObject | GameObject containin the game UI |
| Text | Public text | Start/end UI text element |
| Lvldesc | Public text | Game UI text element |
| Button | Public text | Button text element |
| Level | Private int | Current Level number |

Table 10 - Game controller script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at the start of the application.  Sets level to 0 and disables the game UI |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  If exit button is pressed the application will be closed |
| changeLayover | -Input:  NONE  -Output:  NONE | The function swaps from the Start UI to the game UI |
| changeLevel | -Input:  NONE  -Output:  NONE | This function changes the cameras culling mask to only show the current level. On call it disables the current level and activates the next.  If all levels are beaten it will show the end screen.  Also a level description will be shown |
| getLevel | -Input:  NONE  -Output:  INT | Returns the current level to the calling function as INT |
| Wait | -Input:  NONE  -Output:  IEnumerator | Holds the progression of the script for 5 secounds |

Table 11 - Game controller script functions

### Script::gameStart

This script is responsible the start of “The Gauntlet”.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Camera | Public cameraControl | Camera control script |
| Player | Public Ball | Player control script |
| Transporter | Public transporterControlScript | Transporter control script |
| GC | Public GameController | Game controller script |
| Anim\_Origin | Public Animator | Animator for origin transporter |
| Anim\_Target | Public Animator | Animator for target transporter |

Table 12 - Game start script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called on start of the application.  Sets setting for the start screen UI and locks player control |
| StartNewGame | -Input:  NONE  -Output:  NONE | Set setup into game mode, enabling player control and changes the ui layover |

Table 13 - Game start script functions

### Script::GateControl

Controls the entry gates to level 4.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GateLeft | Public Animator | Animator object for the left entry gate |
| GateRight | Public Animator | Animator object for the right entry gate |

Table 14 - Gate control script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | Starts both gate animations once a collider enters the trigger |

Table 15 - Gate control script functions

### Script::GeneratorControl

Controls the generator animation as well as the animation of the connected component.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Generator | Public animator | Animator object for the generator |
| Supplying | Public animator | Animator for the connected object to the generator |
| Player | Public ball | Ball script |
| Text | Public text | UI text element |
| Active | Private bool | Current state of the generator |

Table 16 - Generator control script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called on start of the application.  Set generator state to false |
| OnTriggerStay | -Input:  Collider  -Output:  IEnumerator | Called every frame while a collider is inside the trigger.  Activates or deactivates the generator depending on its current state when the interaction button is pressed.  Presents messages to the player via the ui text element |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | Sets the ui text element to empty when the collider leaves the trigger |

Table 17 - Generator control script functions

### Script::RespawnScript

Resets the player to a specific point when he fails to complete a task and falls of the level.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Spawn | Public transform | Contains the position of the respawn point |

Table 18 - Respawn script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | Called when a collider enters the trigger.  Sets the position of the collider to the spawn position and reset the gravity to normal |

Table 19 - Respawn script functions

### Script::SimonSays

Main control for the “simon says” game. It handles all incoming input as well as any reaction from the game and generates the sequences for the game.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| ExitRight | Public Animator | Animator object for the right exit gate |
| ExitLeft | Public Animator | Animator for the left exit gate |
| Text | Public text | UI text element |
| Top | Public light | Lighting for top field |
| Right | Public light | Lighting for right field |
| Bottom | Public light | Lighting for bottom field |
| Left | Public light | Lighting for left field |
| Atop | Public AudioSource | Audio for top field |
| Aright | Public AudioSource | Audio for right field |
| Abottom | Public AudioSource | Audio for bottom field |
| Aleft | Public AudioSource | Audio for left field |
| Background | Public AudioSource | Audio for background music |
| Seqnumber | Private int | Current sequence number |
| Currentcount | Private int | Current count in the sequence |
| Seq | Private int[] | Array to store the current sequence |
| Play | Private bool | Bool to show if a game is being played |
| Waittime | Private float | Float number for wait function |

Table 20 - Simon says script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at the start of the application.  Loads the preset for the simon says game |
| getPlay | -Input:  NONE  -Output:  BOOL | Returns to the calling script if a game is being played atm |
| startGame | -Input:  NONE  -Output:  NONE | Starts a new game of siman says. |
| sendChoice | -Input:  INT  -Output:  NONE | If CheckInput function returns true moves on to the next number in the sequence, starts a new sequence or ends the game by opening the gate.  If it returns falls the game will be reset |
| CheckInput | -Input:  INT  -Output:  BOOL | Checks given Input with the current sequence spot and returns true if correct. Otherwise returns false |
| Reset | -Input:  NONE  -Output:  NONE | Resets the game to its default start settings |
| generateNewSeq | -Input:  NONE  -Output:  IEnumerator | This function generates and displays a new sequence for the simon says game after pausing for 3 or 1 second.  The numbers are being generated randomly between the values 1 and 4 making the game different every time it is being played. |

Table 21 - Simon says script functions

### Script::SSChoiceB

Sends a “simon says” choice to the “simon says” script.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SS | Public SimonSays | Simon says script |
| Text | Public text | UI text element |
| Aback | Public AudioSource | Audio source element |
| Back | Public Light | Lighting element |
| waittime | Private float | Time in seconds for wait function |

Table 22 - Simon says back script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  IEnumerator | Called every frame if a collider is inside the trigger.  If a game is currently being played and the interact button is being pressed the light and sound elements are being activated and the choice send to the simon says script. After 1 second wait the light is turned of again |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Resets the UI text element to null if collider leaves the trigger |

Table 23 - Simon says back script functions

### Script::SSChoiceF

Sends a “simon says” choice to the “simon says” script.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SS | Public SimonSays | Simon says script |
| Text | Public text | UI text element |
| Afront | Public AudioSource | Audio source element |
| Front | Public Light | Lighting element |
| waittime | Private float | Time in seconds for wait function |

Table 24 - Simon says front script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  IEnumerator | Called every frame if a collider is inside the trigger.  If a game is currently being played and the interact button is being pressed the light and sound elements are being activated and the choice send to the simon says script. After 1 second wait the light is turned of again |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Resets the UI text element to null if collider leaves the trigger |

Table 25 - Simon says front script functions

### Script::SSChoiceL

Sends a “simon says” choice to the “simon says” script.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SS | Public SimonSays | Simon says script |
| Text | Public text | UI text element |
| Aleft | Public AudioSource | Audio source element |
| Left | Public Light | Lighting element |
| waittime | Private float | Time in seconds for wait function |

Table 26 - Simon says left script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  IEnumerator | Called every frame if a collider is inside the trigger.  If a game is currently being played and the interact button is being pressed the light and sound elements are being activated and the choice send to the simon says script. After 1 second wait the light is turned of again |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Resets the UI text element to null if collider leaves the trigger |

Table 27 - Simon says left script functions

### Script::SSChoiceR

Sends a “simon says” choice to the “simon says” script.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SS | Public SimonSays | Simon says script |
| Text | Public text | UI text element |
| Aright | Public AudioSource | Audio source element |
| Right | Public Light | Lighting element |
| waittime | Private float | Time in seconds for wait function |

Table 28 - Simon says right script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  IEnumerator | Called every frame if a collider is inside the trigger.  If a game is currently being played and the interact button is being pressed the light and sound elements are being activated and the choice send to the simon says script. After 1 second wait the light is turned of again |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Resets the UI text element to null if collider leaves the trigger |

Table 29 - Simon says right script functions

### Script::SSStart

Starts a game of “simon says”.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SS | Public SimonSays | Simon says script |
| Text | Public text | UI text element |
| Background | Public AudioSource | Background music audio source element |

Table 30 - Simon says start script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  NONE  -Output:  NONE | Called every frame a collider is inside the trigger.  If no game is being played and the interaction button is pressed, a new game of simon says is started and the background music is stopped |
| OnTriggerExit | -Input:  NONE  -Output:  NONE | Resets the UI text element to null when a collider leaves the trigger |

Table 31 - Simon says start script functions

### Script::teleportation

The teleportation script moves the player to the target location. It also sends the command to change the current level.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Target | Public transform | Position of the target point |
| Player | Public transform | Position of the player |
| GC | Public GameController | Game Controller script |

Table 32 - teleport script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| transport | -Input:  NONE  -Output:  NONE | If level is less than 5 the player gets repositioned to the target location and the level is changed |

Table 33 - teleport script functions

### Script::transporterControlScript

This script handles the animation of the origin and target transporter.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Anim\_Origin | Public Animator | Animator for the Origin transporter |
| Anim\_Target | Public Animator | Animator for the target transporter |
| Text | Public text | UI text element |

Table 34 - Transporter control script variables

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| OnTriggerStay | -Input:  Collider  -Output:  IEnumerator | Called every frame a collider is inside the trigger.  If the transporter button is pressed, activateTransporter is called and the script is paused for 1 secound |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | Resets the text element to null when collider leaves the trigger |
| activateTransporter | -Input:  Animator, Animator  -Output:  NONE | Activates the animation for the target and origin transporter |

Table 35 - Transporter control script functions

# **Results**

The following picture are taken out from the current version of the game. The show different moments throughout the entire game. Starting with the Star screen, over in game sections to the end screen.

|  |  |
| --- | --- |
| Screenshots | Description |
| Figure 1 - Start screen | The Start Screen of the game |
| Figure 2 - End game screen | The End game screen of the game |
| Figure 3 - Transporter animation | Transporter animation. Here the player is moved from the start screen into level 1 |
| Figure 4 - Generator and UI elements | Generator for the bridge and ui elements |
| Figure 5 - Gravity change mechanic | Start of level 3 demonstrating the gravity change game mechanic |

Table 36 - In game screenshots

# **Annex**

## **Source-Codes**

* Script::Ball

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class Ball : MonoBehaviour { * //external varaible input * public ParticleSystem Anchor; //Particel system of the camera anchor object * public GameController GC; //GameController script * //internal private variables * private float sideMovInput; //Input for side movement * private float forwardMovInput; //input for fov/back movement * private Rigidbody rig; //riged body of the object * private bool charged; //if ball is charged (true, false) * // Use this for initialization * void Start () { * charged = false; //set charged to false * rig = GetComponent<Rigidbody>(); //get Rig Body component of gameobject * } * // Update is called once per frame * void Update () { * forwardMovInput = Input.GetAxis("MoveVer") \* 150; //read input from vertical axis \*150 (up down move) * sideMovInput = Input.GetAxis("MoveHor") \* 150; //read input from horizontal axis \*150(left right mov) * if (forwardMovInput!=0) //if forward movment is not 0 * { * rig.AddTorque(Camera.main.transform.right \* forwardMovInput); //apply torgue to teh rig body of the ball in relation to camera view * } * if (sideMovInput != 0) //if side movement is not 0 * { * rig.AddTorque(Camera.main.transform.forward \* -sideMovInput); //apply torgue to teh rig body of the ball in relation to camera view * } * if ((Input.GetButtonDown("Jump"))&&!charged) //if jump is pressed and ball is not charged * { * rig.AddForce(Camera.main.transform.up + new Vector3(0,12,0),ForceMode.Impulse); //apply force to teh ball to it jump * } * if ((Input.GetButtonDown("Drop")) && charged) //if drop is pressed and player is charged * { * charged = false; //change current charge status to false to remove charge * } * if (GC.getLevel()==3) //if player is in level 3 * { * if(Input.GetAxis("GravVer") > 0 || Input.GetKeyDown("1")) //get key input * { * Physics.gravity = Camera.main.transform.up+new Vector3(0, 9.8f, 0); //changing gravity direction to upwards * } * if (Input.GetAxis("GravVer") < 0 || Input.GetKeyDown("2")) //get key input * { * Physics.gravity = Camera.main.transform.up + new Vector3(0, -9.8f, 0); //changing gravity direction to downwards * } * if(Input.GetAxis("GravHor") < 0 || Input.GetKeyDown("3")) //get key input * { * Physics.gravity = Camera.main.transform.right \* -9.8f; //changing gravity direction to left in relation to the camera view * } * if(Input.GetAxis("GravHor") > 0 ||Input.GetKeyDown("4")) //get key input * { * Physics.gravity = Camera.main.transform.right \* 9.8f; //changing gravity direction to right in relation to the camera view * } * } * if(charged) //if ball is charged * { * Anchor.enableEmission=true; //play particle system of the ball * } * else * { * Anchor.enableEmission = false; //stop particle system of the ball * } * } * //fucntion to charge the ball to deliver charge to generator * public void charge() * { * charged = !charged; //set charge to invert of current value * } * //getter for charge variable * public bool readCharge() * { * return charged; //return charge value * } * } |

Code 1 - Ball Script

* Script::Bridge

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class Bridge : MonoBehaviour { * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //trigger event when entered * private void OnTriggerEnter(Collider obj) * { * obj.transform.parent = gameObject.transform; //set collider object to child of bridge trigger * } * //trigger event when left * private void OnTriggerExit(Collider obj) * { * obj.transform.parent = null; //set collider to have no parent * obj.transform.localScale = new Vector3(4, 4, 4); * } * } |

Code 2 - Bridge Script

* Script::cameraControl

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class cameraControl : MonoBehaviour { * //local private variables * private float camSideMov; //camera side movement value * private float camUpMov; //camera up movement value * private GameObject player; //gameobject of the player * private GameObject anchor; //gameobject of the anchor object * // Use this for initialization * void Start () { * player = GameObject.FindGameObjectWithTag("Player"); //find gameobject of the player * anchor = GameObject.FindGameObjectWithTag("Anchor"); //find gameobject of the anchor * } * // Update is called once per frame * void Update () { * camSideMov = Input.GetAxis("CamHor"); //get input from X axis * camUpMov = Input.GetAxis("CamVer"); //get input from Y axis * anchor.transform.position = player.transform.position; //set anchor position to player position * Camera.main.transform.RotateAround(anchor.transform.position, Camera.main.transform.up, camSideMov); //rotate the camera around the anchor object * Camera.main.transform.RotateAround(anchor.transform.position, Camera.main.transform.right, camUpMov); //rotate the camera around the anchor object * Camera.main.transform.LookAt(anchor.transform.position); //forca camera to look at anchor in relation to world grid * } * } |

Code 3 - Camera Control Script

* Script::ChargePlayer

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class ChargePlayer : MonoBehaviour { * //external variable input * public Ball player\_script; //Script attached to the player object * public Text text; //UI Text element * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //trigger event when collider inside * private IEnumerator OnTriggerStay() * { * if(!player\_script.readCharge()) //if ball is not charged * { * text.text = "Press 'P' or 'X-Button' to charge the ball"; //change text of ui object * if (Input.GetButtonDown("Interact")) //get key input * { * player\_script.charge(); //charge ball * yield return new WaitForSeconds(1); //debounce/wait for 1 secound * } * } * else * text.text = "Ball is already charged"; //change text of ui object * } * //trigger event when leaving * private void OnTriggerExit() * { * text.text = ""; //set text to empty * } * } |

Code 4 - Charge Player Script

* Script::GameController

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class GameController : MonoBehaviour { * //external input variable * public cameraControl cam; //camera control script * public Ball Player; //player control script * public GameObject StartCanvas; //menu ui elements * public GameObject GameCanvas; //game ui elements * public Text text; //text element * public Text lvldesc; //level description text element * public Text button; //button text element * //private variables * private int level; //current level being played * // Use this for initialization * void Start () { * level = 0; //set level value to 0 * GameCanvas.SetActive(false); //set game ui to off * } * // Update is called once per frame * void Update () { * if (Input.GetButtonDown("Quit")) //get key input * { * Application.Quit(); //close application * } * } * //change UI layover to game ui * public void changeLayover() * { * StartCanvas.SetActive(false); //turn off menu ui * GameCanvas.SetActive(true); //turn on game ui * } * //change level * public void changeLevel() * { * if (level< 4&&Cursor.visible==false) //if level is less then 4 and cursor is off * { * Camera.main.cullingMask ^= 1 << LayerMask.NameToLayer("Lvl" + level); //hide current level * level++; //increas level * Camera.main.cullingMask ^= 1 << LayerMask.NameToLayer("Lvl" + level); //show next level * switch(level) //select depending on level value * { * case 0: //level 0 * lvldesc.text = "Objective: Can you beat ‘The GAUNTLET’"; //change text object * break; * case 1: //level 1 * lvldesc.text = "Objective: Reach the transporter"; //change text object * break; * case 2: //level 2 * lvldesc.text = "Objective: Find a way to reach the other side"; //change text object * break; * case 3: //level 3 * lvldesc.text = "Objective: Messing with Gravity can be dangerousy)"; //change text object * break; * case 4: //level 4 * lvldesc.text = "Objective: Lets play Simon Says.\r\n Beat the game to advance"; //change text object * break; * } * } * else if(level>=4) //if level is 4 or higher * { * text.text = "CONGRATULATIONS! \r\n You have beaten the gauntlet."; //set text to show * button.text = "Play Again."; //set button text * GameCanvas.SetActive(false); //turn off game ui * StartCanvas.SetActive(true); //turn on menu ui * Camera.main.cullingMask ^= 1 << LayerMask.NameToLayer("Lvl" + level); //turn off curent level * level=0; //set level to 0 * Camera.main.cullingMask ^= 1 << LayerMask.NameToLayer("Lvl" + level); //turn on level 0 * Player.enabled = false; //disable player control * cam.enabled = false; //disable camera control * Cursor.visible = true; //show cursor * } * } * //getter for level variable * public int getLevel() * { * return level; //return level value * } * //wait function * private IEnumerator wait() * { * yield return new WaitForSeconds(5); //wait for 1 secound * } * } |

Code 5 - Game controller script

* Script::gameStart

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class gameStart : MonoBehaviour { * //external variable input * public cameraControl Camera; //camera control script * public Ball Player; //player control script * public transporterControlScript transporter; //transporter script * public GameController GC; //game controller script * public Animator anim\_Origin; //Animator Origin * public Animator anim\_Target; //Animator destination * // Use this for initialization * void Start () { * Cursor.visible=true; //set cursor to visible * Camera.enabled = false; //turn off camera script * Player.enabled = false; //turn off player script * Cursor.lockState = CursorLockMode.Confined; //lock cursor to game window * } * // Update is called once per frame * void Update () { * } * //called when new game is started * public void StartNewGame() * { * Cursor.visible = false; //set cursor to invisible * Camera.enabled = true; //enable camera control * Player.enabled = true; //enable player control * GC.changeLayover(); //change ui layover * } * } |

Code 6 - Game start script

* Script::GateControl

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class GateControl : MonoBehaviour { * //External input placeholder * public Animator GateLeft; //animator for left gate * public Animator GateRight; //animatro of right gate * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //trigger when enterign * void OnTriggerEnter(Collider obj) * { * GateLeft.SetTrigger("open"); //trigger left gate to open * GateRight.SetTrigger("open"); //trigger right gate to open * } * } |

Code 7 - Gate control script

* Script::GeneratorControl

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class GeneratorControl : MonoBehaviour { * //external Interface Variables * public Animator generator; //Animator generator * public Animator supplying; //Animator target to power * public Ball player; //player control script * public Text text; //ui text element * //internalprivate variables * private bool active; //generator current state * // Use this for initialization * void Start () { * active = false; //set active to false * } * // Update is called once per frame * void Update () { * } * //trigger input event to power/unpoer generator * public IEnumerator OnTriggerStay(Collider obj) * { * if (!active && player.readCharge())//generator is not active and ball is charged * { * text.text = "Press 'P' or 'X-Button' to charge the generator"; //change text element * if (Input.GetButtonDown("Interact")) //on key input * { * active = true; //set generator to active * player.charge(); //set player charge * generator.SetTrigger("power"); //trigger animation generator * supplying.SetBool("power\_supply", true); //start animation bridge * yield return new WaitForSeconds(1); //wait for 1 secound * } * } * else if (active && !player.readCharge())//generator is active and ball is uncharged * { * text.text = "Press 'P' or 'X-Button' to decharge the generator"; //change text element * if (Input.GetButtonDown("Interact")) //on key input * { * active = false; //set generator to inactive * player.charge(); //set player charge * generator.SetTrigger("unpower"); //trigger animation generator * supplying.SetBool("power\_supply", false); //stop animation bridge * yield return new WaitForSeconds(1); //wait for 1 secound * } * } * else if (!active && !player.readCharge()) //if not active and player is not charged * { * text.text = "Find a charge to activate the generator"; //change text element * } * else if (active && player.readCharge()) //if active and player is charged * { * text.text = "Already charged - Cannot take on another charge"; //change text element * } * } * //trigger on leave * public void OnTriggerExit(Collider obj) * { * text.text = ""; //change text element * } * } |

Code 8 - Generator control script

* Script::RespawnScript

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class RespawnScript : MonoBehaviour { * //External input placeholder * public Transform Spawn; //respawn point on Character death * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //trigger if player collides with plane * private void OnTriggerEnter(Collider obj) * { * Physics.gravity = new Vector3(0, -9.8f, 0); //set gravity to down * obj.transform.position=Spawn.position; //Reset player to Spawn position * } * } |

Code 9 - Respawn script

* Script::SimonSays

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SimonSays : MonoBehaviour { * //external input variable * public Animator ExitRight; //Animator from right exit gate object * public Animator ExitLeft; //Animator from left exit gate object * public Text text; //ui text element * public Light top; //Spotlight object for top field * public Light right; //Spotlight object for right field * public Light bottom; //Spotlight object for bottom field * public Light left; //Spotlight object for left field * public AudioSource Atop; //sound source for top field * public AudioSource Aright; //sound source for right field * public AudioSource Abottom; //sound source for bottom field * public AudioSource Aleft; //sound source for left field * public AudioSource background; //background music * //private variables * private int seqnumber; //current sequence number * private int currentcount; //number of selected choices * private int[] seq = new int[8]; //stored sequence to eb repeated * private bool play; //if game is being played * private float waittime; //waittime for wait function * // Use this for initialization * void Start () { * seqnumber = 1; //set start sequence to 1 * currentcount = 0; //set current count to 0 * play = false; //set play false * } * // Update is called once per frame * void Update () { * } * //getter for play variable * public bool getPlay() * { * return play; //return play value * } * //function to start a new game * public void startGame() * { * play = true; //set play true * waittime = 1; //set waittime 1 * seqnumber = 1; //set sequence to 1 * currentcount = 0; //set current count to 0 * StartCoroutine(generateNewSeq()); //generate a new sequence * } * //function to handle player choice * public void sendchoice(int choice) * { * if (CheckInput(choice))//if player choice is correct * { * currentcount++;//increase current count * if (currentcount == seqnumber) //if current round has been completted * { * if (seqnumber == 5) //if five rounds have been completted * { * text.text = "Congratulations! The final Gate is now open"; //change text element * background.Play(); * ExitLeft.SetTrigger("open"); //trigger left gate to open * ExitRight.SetTrigger("open"); //trigger right get to open * play = false; //set play to false * } * else * { * seqnumber++; //increase sequence number * currentcount = 0; //set current count to 0 * StartCoroutine(generateNewSeq()); //generate a new sequence * } * } * } * else * { * text.text = "Wrong sequence has been entered. The game will start again."; //change text element * reset(); //reset the game * } * } * //function to check player input * private bool CheckInput(int input) * { * if(seq[currentcount]==input) //if choice matches the given input * return true; //return true * else * return false; //return false * } * //function to reset the game * private void reset() * { * seq = new int[8]; //create new sequence array * seqnumber =1; //set sequence number to 1 * currentcount=0; //set current count to 0 * play = false; //set play to false * } * //function to generate a new game sequence * private IEnumerator generateNewSeq() * { * if(seqnumber>1) //if not first round * yield return new WaitForSeconds(3); //wait 3 secounds * else * yield return new WaitForSeconds(1); //wait 1 secound * for (int i=0; i<seqnumber; i++) //run through the entire given sequence * { * seq[i] = Random.Range(1, 5); //create random number between 1 and 4 * switch(seq[i]) //decide what to do in relation to generated number * { * case 1: //if 1 * top.enabled = true; //enable top field lighting * Atop.Play(); //play top sound * yield return new WaitForSeconds(waittime); //wait for x secounds * top.enabled = false; //disable top field lighting * break; * case 2: //if 2 * right.enabled = true; //enable right lighting * Aright.Play(); //play right sound * yield return new WaitForSeconds(waittime); //wait for x secounds * right.enabled = false; //disable right lighting * break; * case 3: //if 3 * bottom.enabled = true; //enable bottom lighting * Abottom.Play(); //play bottom sound * yield return new WaitForSeconds(waittime); //wait for x secounds * bottom.enabled = false; //disable bottom lighting * break; * case 4: //if 4 * left.enabled = true; //enable left lighting * Aleft.Play(); //play left sound * yield return new WaitForSeconds(waittime); //wait for x secounds * left.enabled = false; //disable left lighting * break; * } * } * } * } |

Code 10 - Simon says script

* Script::SSChoiceB

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SimonSays : MonoBehaviour { * //external input variable * public Animator ExitRight; //Animator from right exit gate object * public Animator ExitLeft; //Animator from left exit gate object * public Text text; //ui text element * public Light top; //Spotlight object for top field * public Light right; //Spotlight object for right field * public Light bottom; //Spotlight object for bottom field * public Light left; //Spotlight object for left field * public AudioSource Atop; //sound source for top field * public AudioSource Aright; //sound source for right field * public AudioSource Abottom; //sound source for bottom field * public AudioSource Aleft; //sound source for left field * public AudioSource background; //background music * //private variables * private int seqnumber; //current sequence number * private int currentcount; //number of selected choices * private int[] seq = new int[8]; //stored sequence to eb repeated * private bool play; //if game is being played * private float waittime; //waittime for wait function * // Use this for initialization * void Start () { * seqnumber = 1; //set start sequence to 1 * currentcount = 0; //set current count to 0 * play = false; //set play false * } * // Update is called once per frame * void Update () { * } * //getter for play variable * public bool getPlay() * { * return play; //return play value * } * //function to start a new game * public void startGame() * { * play = true; //set play true * waittime = 1; //set waittime 1 * seqnumber = 1; //set sequence to 1 * currentcount = 0; //set current count to 0 * StartCoroutine(generateNewSeq()); //generate a new sequence * } * //function to handle player choice * public void sendchoice(int choice) * { * if (CheckInput(choice))//if player choice is correct * { * currentcount++;//increase current count * if (currentcount == seqnumber) //if current round has been completted * { * if (seqnumber == 5) //if five rounds have been completted * { * text.text = "Congratulations! The final Gate is now open"; //change text element * background.Play(); * ExitLeft.SetTrigger("open"); //trigger left gate to open * ExitRight.SetTrigger("open"); //trigger right get to open * play = false; //set play to false * } * else * { * seqnumber++; //increase sequence number * currentcount = 0; //set current count to 0 * StartCoroutine(generateNewSeq()); //generate a new sequence * } * } * } * else * { * text.text = "Wrong sequence has been entered. The game will start again."; //change text element * reset(); //reset the game * } * } * //function to check player input * private bool CheckInput(int input) * { * if(seq[currentcount]==input) //if choice matches the given input * return true; //return true * else * return false; //return false * } * //function to reset the game * private void reset() * { * seq = new int[8]; //create new sequence array * seqnumber =1; //set sequence number to 1 * currentcount=0; //set current count to 0 * play = false; //set play to false * } * //function to generate a new game sequence * private IEnumerator generateNewSeq() * { * if(seqnumber>1) //if not first round * yield return new WaitForSeconds(3); //wait 3 secounds * else * yield return new WaitForSeconds(1); //wait 1 secound * for (int i=0; i<seqnumber; i++) //run through the entire given sequence * { * seq[i] = Random.Range(1, 5); //create random number between 1 and 4 * switch(seq[i]) //decide what to do in relation to generated number * { * case 1: //if 1 * top.enabled = true; //enable top field lighting * Atop.Play(); //play top sound * yield return new WaitForSeconds(waittime); //wait for x secounds * top.enabled = false; //disable top field lighting * break; * case 2: //if 2 * right.enabled = true; //enable right lighting * Aright.Play(); //play right sound * yield return new WaitForSeconds(waittime); //wait for x secounds * right.enabled = false; //disable right lighting * break; * case 3: //if 3 * bottom.enabled = true; //enable bottom lighting * Abottom.Play(); //play bottom sound * yield return new WaitForSeconds(waittime); //wait for x secounds * bottom.enabled = false; //disable bottom lighting * break; * case 4: //if 4 * left.enabled = true; //enable left lighting * Aleft.Play(); //play left sound * yield return new WaitForSeconds(waittime); //wait for x secounds * left.enabled = false; //disable left lighting * break; * } * } * } * } |

Code 11 - Simon says bottom field script

* Script::SSChoiceF

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SSChoiceF : MonoBehaviour { * //external Interface Variables * public SimonSays SS; //Simon Says Script * public Text text; //UI Text element * public AudioSource Afront; //audio source * public Light front; //lighting * //private variables * private float waittime = 1; //set waittime to 1 * // Use this for initialization * void Start() * { * } * // Update is called once per frame * void Update() * { * } * //trigger while collider is inside * public IEnumerator OnTriggerStay() * { * if (SS.getPlay()) //check if game is being played * { * text.text = "Press 'P' or 'X-Button' to select choice"; //change text object * if (Input.GetButtonDown("Interact")) //on key input * { * front.enabled = true; //enable lighting * Afront.Play(); //play sound * SS.sendchoice(1); //send choice to simon says script * yield return StartCoroutine(wait()); //wait * front.enabled = false; //disable lighting * } * } * } * //trigger on exit * public void OnTriggerExit() * { * text.text = ""; //change text object * } * //wait function * private IEnumerator wait() * { * yield return new WaitForSeconds(waittime); //wait for x secounds * } * } |

Code 12 - Simon says front field script

* Script::SSChoiceL

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SSChoiceL : MonoBehaviour { * //external Interface Variables * public SimonSays SS; //Simon Says Script * public Text text; //UI Text element * public AudioSource Aleft; //audio source * public Light left; //lighting * //private variables * private float waittime = 1; //set waittime to 1 * // Use this for initialization * void Start() * { * } * // Update is called once per frame * void Update() * { * } * //trigger while collider is inside * public IEnumerator OnTriggerStay() * { * if (SS.getPlay()) //if game is being played * { * text.text = "Press 'P' or 'X-Button' to select choice"; //change text object * if (Input.GetButtonDown("Interact")) //on key input * { * left.enabled = true; //enable lighting * Aleft.Play(); //play sound * SS.sendchoice(4); //send choice to simon says script * yield return StartCoroutine(wait()); //wait * left.enabled = false; //disable lighting * } * } * } * //trigger on exit * public void OnTriggerExit() * { * text.text = ""; //change text object * } * //wait function * private IEnumerator wait() * { * yield return new WaitForSeconds(waittime); //wait for x secounds * } * } |

Code 13 - Simon says left field script

* Script::SSChoiceR

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SSChoiceR : MonoBehaviour { * //external Interface Variables * public SimonSays SS; //Simon Says Script * public Text text; //UI Text element * public AudioSource Aright; //audio source * public Light right; //lighting * //private variables * private float waittime = 1; //set waittime to 1 * // Use this for initialization * void Start() * { * } * // Update is called once per frame * void Update() * { * } * //trigger while collider inside * public IEnumerator OnTriggerStay() * { * if (SS.getPlay()) //if game is being played * { * text.text = "Press 'P' or 'X-Button' to select choice"; //change text object * if (Input.GetButtonDown("Interact")) //on key input * { * right.enabled = true; //enable lighting * Aright.Play(); //play sound * SS.sendchoice(2); //send choice to simon says script * yield return StartCoroutine(wait()); //wait * right.enabled = false; //disable lighting * } * } * } * //trigger on exit * public void OnTriggerExit() * { * text.text = ""; //change text object * } * //wait function * private IEnumerator wait() * { * yield return new WaitForSeconds(waittime); //wait for x secounds * } * } |

Code 14 - Simon says right field script

* Script::SSStart

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class SSStart : MonoBehaviour { * //external Interface Variables * public SimonSays SS; //Simon Says Script * public Text text; //UI Text element * public AudioSource background; //background audio source * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //trigger while collider inside * private void OnTriggerStay() * { * if (!SS.getPlay()) //if no game is being played * { * text.text = "Press 'P' or 'X-Button' to start playing Simon Says"; //change text object * if (Input.GetButtonDown("Interact")) //get key input * { * background.Stop(); //stop background music * SS.startGame(); //start game of simon says * } * } * else * text.text = ""; //change text object * } * //trigger on exit * private void OnTriggerExit() * { * text.text = ""; //change text object * } * } |

Code 15 - Simon says start script

* Script::teleportation

|  |
| --- |
| * using UnityEngine; * using System.Collections; * public class teleportation : MonoBehaviour { * //external Interface Variables * public Transform target; //destination tranform component * public Transform player; //PLayer transform component * public GameController GC; //game controller script * // Use this for initialization * void Start () { * } * // Update is called once per frame * void Update () { * } * //Transport function for player movement * public void transport() * { * if (GC.getLevel() < 5) //if level is below 5 * { * player.position = target.position + new Vector3(0, 5, 0); //move player to target destination * GC.changeLevel(); //change level * } * } * } |

Code 16 - teleportation script

* Script::transporterControlScript

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * using System.Collections; * public class transporterControlScript : MonoBehaviour { * //external Interface Variables * public Animator anim\_Origin; //Animator Origin * public Animator anim\_Target; //Animator destination * public Text text; //ui text object * // Use this for initialization * void Start() * { * } * // Update is called once per frame * void Update() * { * } * //Trigger Enter Event(change to trigger stay and call button press check) * public IEnumerator OnTriggerStay(Collider obj) * { * text.text = "Press 'T' or 'triangle-button' to activate transporter"; //change text object * if (Input.GetButtonDown("Teleport")) //on KeyPress t when inside the trigger zone * { * activateTransporter(anim\_Target, anim\_Origin); //call transporter animation * yield return new WaitForSeconds(1); //wait for 1 secound * } * } * //trigger on exit * public void OnTriggerExit(Collider obj) * { * text.text = ""; //change text object * } * //activate transporter animation * public void activateTransporter(Animator target, Animator origin) * { * target.SetTrigger("activate"); //play animation at target * origin.SetTrigger("activate"); //play animation at origin * } * } |

Code 17 - Transporter control script

## **Table of Code segments**

[Code 1 - Ball Script 22](#_Toc464763284)

[Code 2 - Bridge Script 22](#_Toc464763285)

[Code 3 - Camera Control Script 23](#_Toc464763286)

[Code 4 - Charge Player Script 24](#_Toc464763287)

[Code 5 - Game controller script 26](#_Toc464763288)

[Code 6 - Game start script 27](#_Toc464763289)

[Code 7 - Gate control script 28](#_Toc464763290)

[Code 8 - Generator control script 29](#_Toc464763291)

[Code 9 - Respawn script 30](#_Toc464763292)

[Code 10 - Simon says script 33](#_Toc464763293)

[Code 11 - Simon says bottom field script 37](#_Toc464763294)

[Code 12 - Simon says front field script 38](#_Toc464763295)

[Code 13 - Simon says left field script 39](#_Toc464763296)

[Code 14 - Simon says right field script 41](#_Toc464763297)

[Code 15 - Simon says start script 41](#_Toc464763298)

[Code 16 - teleportation script 42](#_Toc464763299)

[Code 17 - Transporter control script 43](#_Toc464763300)

## **Table of Figures**

[Figure 1 - Start screen 18](#_Toc464763279)

[Figure 2 - End game screen 18](#_Toc464763280)

[Figure 3 - Transporter animation 18](#_Toc464763281)

[Figure 4 - Generator and UI elements 19](#_Toc464763282)

[Figure 5 - Gravity change mechanic 19](#_Toc464763283)

## **Table of Tables**

[Table 1- Game Controls 5](#_Toc464763243)

[Table 2 - Ball script variables 6](#_Toc464763244)

[Table 3 - Ball script functions 6](#_Toc464763245)

[Table 4 - Bridge script variables 6](#_Toc464763246)

[Table 5 - Bridge script functions 7](#_Toc464763247)

[Table 6 - Camera control script variables 7](#_Toc464763248)

[Table 7 - Camera control script functions 7](#_Toc464763249)

[Table 8 - Charge player script variables 7](#_Toc464763250)

[Table 9 - Charge player script functions 8](#_Toc464763251)

[Table 10 - Game controller script variables 8](#_Toc464763252)

[Table 11 - Game controller script functions 9](#_Toc464763253)

[Table 12 - Game start script variables 9](#_Toc464763254)

[Table 13 - Game start script functions 9](#_Toc464763255)

[Table 14 - Gate control script variables 10](#_Toc464763256)

[Table 15 - Gate control script functions 10](#_Toc464763257)

[Table 16 - Generator control script variables 10](#_Toc464763258)

[Table 17 - Generator control script functions 11](#_Toc464763259)

[Table 18 - Respawn script variables 11](#_Toc464763260)

[Table 19 - Respawn script functions 11](#_Toc464763261)

[Table 20 - Simon says script variables 12](#_Toc464763262)

[Table 21 - Simon says script functions 12](#_Toc464763263)

[Table 22 - Simon says back script variables 13](#_Toc464763264)

[Table 23 - Simon says back script functions 13](#_Toc464763265)

[Table 24 - Simon says front script variables 13](#_Toc464763266)

[Table 25 - Simon says front script functions 14](#_Toc464763267)

[Table 26 - Simon says left script variables 14](#_Toc464763268)

[Table 27 - Simon says left script functions 14](#_Toc464763269)

[Table 28 - Simon says right script variables 15](#_Toc464763270)

[Table 29 - Simon says right script functions 15](#_Toc464763271)

[Table 30 - Simon says start script variables 15](#_Toc464763272)

[Table 31 - Simon says start script functions 16](#_Toc464763273)

[Table 32 - teleport script variables 16](#_Toc464763274)

[Table 33 - teleport script functions 16](#_Toc464763275)

[Table 34 - Transporter control script variables 16](#_Toc464763276)

[Table 35 - Transporter control script functions 17](#_Toc464763277)

[Table 36 - In game screenshots 19](#_Toc464763278)

## **References**

Content used from the Unity asset store:

* Metal texture pack
* Island
* Jukebox
* Fences
* Purple Nebula
* 8-Bit Sounds
* Persuit of death
* Lightning
* Stone wall