**Respawn Script Tutorial**

void Update()

{

if (Input.GetKeyDown(KeyCode.R))

Application.LoadLevel(0);

}

1. Create a new 3D Unity project.
2. Go to GameObject on the top of the screen -> 3D Object -> Cube – We will use this as the base for the sphere.
3. Set the cubes X and Z scale to 5 – This is so that the base is bigger to work with.
4. Go to GameObject on the top of the screen -> 3D Object -> Sphere – We will use this object to test if the code works.
5. Make sure the Sphere is at a nice distance above the cube – This is so we can use the gravity we will add in the next three steps.
6. On the spheres inspector, click Add Component and search for Rigidbody – This will allow the ball to move in the scene and be affected by Unity’s physics system.
7. Make sure that this Use Gravity is ticked – This will allow the sphere to fall with gravity – We using this so we can test that the respawn script is working.
8. Click the play button – This is to test that the gravity is working.
9. On assets tab, right click and select Create -> C# Script and name it something related to the code we will write. i.e Respawn. – This will create a default code file that we will add our code to.
10. Open the code file by double clicking the file in the assets tab.
11. Remove all code from lines 7 – 17 – We will be replacing it with the respawn script.
12. On line 7 type “ void Update()” – This will constantly update the code so that it can be used at all times when the game is being played.
13. On line 8 add an open squiggly bracket “ {“– This will open up a new section of the code to work under void update.
14. On line 9 type “ if (Input.GetKeyDown(KeyCode.R))” – This will run the all the code underneath in the section when it senses the key “r” has been pressed.
15. On line 10 type “ Application.LoadLevel(0);” – This will make the scene begin back at the start.
16. On line 12 add a close squiggly bracket “ }” – This will close of the code section.
17. Save the code and return to Unity.
18. Drag the script onto any of the objects, i.e. the sphere, in the Unity scene – This is so that the script can run on the scene.
19. Click play and then press r – this is to test if the code works – if sphere respawns above the cube and falls every time you press the key “r”, then the code is working.
20. Save the Unity project – So you don’t lose the progress.

**Jumping Script Tutorial**

public bool onGround;

private Rigidbody rb;

// Start is called before the first frame update

void Start()

{

onGround = true;

rb = GetComponent<Rigidbody>();

}

// Update is called once per frame

void Update()

{

if(onGround)

{

if(Input.GetButtonDown("Jump"))

{

rb.velocity = new Vector3(0f, 5f, 0f);

onGround = false;

}

}

}

void OnCollisionEnter(Collision other)

{

if(other.gameObject.CompareTag("Floor"))

{

onGround = true;

}

}

1. Start a new Unity file.
2. Go to GameObject on the top of the screen -> 3D Object -> Plane – We will use this as the floor for the player to jump on.
3. Click on the plane then click on the tag tab and select add tag.
4. Click on the + then call it Floor – This will allow specific code to play when in contact with a specific object.
5. Click on the plane again then click on the tag tab and select Floor.
6. Make sure there is a box collider on the plane so that it can interact with the player.
7. Go to GameObject -> 3D Object -> Capsule – We will use this as the object to jump with.
8. Move the object slightly upwards above the plane.
9. Go to Add Component and add Rigidbody – This will allow the object to move with code.
10. Expand the Constraints section and select X and Z on Freeze Rotation – This will stop the object from falling onto its sides.
11. Go to the assets window then right click and select Create -> C# Script then call it Jumping.
12. Open up the Jumping script.
13. From lines 8 to 37 add this script:

public bool onGround; - This will create as true/false variable.

private Rigidbody rb; - This will make a Rigidbody varial to allow the object to move.

// Start is called before the first frame update

void Start() – This will play the code inside its section at the start of the session.

{

onGround = true; This will set onGround variable to true.

rb = GetComponent<Rigidbody>(); - This will activate the Rigidbody functionality.

}

// Update is called once per frame

void Update() – This will play the code inside its section constantly throughout the session.

{

if(onGround) - This will play the following section of code if onGround equals true.

{

if(Input.GetButtonDown("Jump")) – This will play the following section of code if the space button is held down.

{

rb.velocity = new Vector3(0f, 5f, 0f); - This will move the object upwards.

onGround = false; - This will set onGround to false, so that the code the object will not be active again.

}

}

}

void OnCollisionEnter(Collision other) – This will play the section of the code if the object collides with an object with a box collider.

{

if(other.gameObject.CompareTag("Floor")) – This will play the following section of code if it comes in contact with an object with the Floor tag.

{

onGround = true; - This will set the onGround variable to true, allowing the jump animation to work again.

}

}

1. Save and go back to unity.
2. Add the code to the capsule by dragging the Jumping script from assets to the bottom of the inspector window.
3. Press play to test if the code works.

**Camera Switch Script**

{

public GameObject cam1;

public GameObject cam2;

public GameObject cam3;

public GameObject cam4;

// Update is called once per frame

void Update()

{

if(Input.GetButtonDown("Key1"))

{

cam1.SetActive(true);

cam2.SetActive(false);

cam3.SetActive(false);

cam4.SetActive(false);

}

if (Input.GetButtonDown("Key2"))

{

cam1.SetActive(false);

cam2.SetActive(true);

cam3.SetActive(false);

cam4.SetActive(false);

}

if (Input.GetButtonDown("Key3"))

{

cam1.SetActive(false);

cam2.SetActive(false);

cam3.SetActive(true);

cam4.SetActive(false);

}

if (Input.GetButtonDown("Key4"))

{

cam1.SetActive(false);

cam2.SetActive(false);

cam3.SetActive(false);

cam4.SetActive(true);

}

}

}

1. Create a new Unity file.
2. Set up a simple scene with a plane and some random obejcts by going to GaemObject -> 3D Object – This is simply so it would become easier to see the camera move about – Do not use quad as that object is invisible.
3. Go to GameObject -> Camera to create a new camera in the scene and press CTRL + D to duplicate it – Do this four times – This is because we will be switching between four different characts.
4. Move the cameras to the four sides of the plane and rotate them toward the centre of the plane – press W to position mode and e to rotate mode – use the green circle to rotate on the y axis – This will make it possible to see that the camera has switched in the game view.
5. Rename the cameras Camera 1, Camera 2, Camera 3, Camera 4 – This will make identifying the camera obejcts easier.
6. Then go to edit -> Project Setting -> Input Manager – This is so we can have controlls for specific keys.
7. Change the size to 22 to add four more controlls – This will duplicate the last controll four times.
8. Change the name of the last four controlls to Key1, Key2, Key3, Key4 – This will make identifying controlls easier and wont conflict with each other.
9. Change the Positive Button in each Key controll to the same number used in the naem – This will make it so that the controll will only work with that key – You can change the Positive Button to any key you like if you want.
10. Go to the Project Tab -> Asset folder and right click -> Create -> C# Script and call it CameraSwitching.
11. Input this script:

{

public GameObject cam1; - This will create a variable for the first camera.

public GameObject cam2; - This will create a variable for the second camera.

public GameObject cam3; - This will create a variable for the third camera.

public GameObject cam4; - This will create a variable for the fourth camera.

// Update is called once per frame

void Update() – This will continually run the code throughout the whole game session.

{

if(Input.GetButtonDown("Key1")) – This will activate the next section of the code when Key1’s Positive Button is pressed.

{

cam1.SetActive(true); - This will show the first camera.

cam2.SetActive(false); - This will hide the second camera.

cam3.SetActive(false); - This will hide the third camera.

cam4.SetActive(false); - This will hide the fourth camera.

}

if (Input.GetButtonDown("Key2")) – This will activate the next section of the code when Key2’s Positive Button is pressed.

{

cam1.SetActive(false); - This will hide the first camera.

cam2.SetActive(true); - This will show the second camera.

cam3.SetActive(false); - This will hide the third camera.

cam4.SetActive(false); - This will hide the fourth camera.

}

if (Input.GetButtonDown("Key3")) – This will activate the next section of the code when Key3’s Positive Button is pressed.

{

cam1.SetActive(false); - This will hide the first camera.

cam2.SetActive(false); - This will hide the second camera.

cam3.SetActive(true); - This will show the third camera.

cam4.SetActive(false); - This will hide the fourth camera.

}

if (Input.GetButtonDown("Key4")) – This will activate the next section of the code when Key4’s Positive Button is pressed.

{

cam1.SetActive(false); - This will hide the first camera.

cam2.SetActive(false); - This will hide the second camera.

cam3.SetActive(false); - This will hide the third camera.

cam4.SetActive(true); - This will show the fourth camera.

}

}

}

1. Save the code and head back to Unity.
2. Drag the script onto any of the objects on the screen except for the cameras – This will allow for the script to run on the scene – Assigning the script to a camera will stop the code from running when that camera is hidden.
3. Go to the object the script is assign to -> Navigate to the script’s section in the Inspector tab.
4. Drag the correct Camera objects to the cam vairable with the same number – This will assign the camera objects to the vairables.
5. Hold shift to multi-select Camera objects 2, 3 and 4 and untick the box at the top left of the Inspector tab – This will hide cameras 2, 3 and 4 so the game only starts with Camera 1.
6. Save the Unity project and click play to test if the game works.

**Rotating Script**

{

float speed = 50.0f;

// Update is called once per frame

void Update()

{

transform.Rotate(Vector3.up \* speed \* Time.deltaTime);

}

}

1. Create a new Unity file.
2. Got to GameObject -> 3D Object -> Cube – This will be the object that we will make rotate with code.
3. Go to the assets file in project and right click -> Create -> C# Script, call it RotateObject and open the file.
4. Input the code:

{

float speed = 50.0f; - This will create a number variable that will tell how fast the object will rotate.

// Update is called once per frame

void Update() – This will constantly run the code within its section throughout the entire game session.

{

transform.Rotate(Vector3.up \* speed \* Time.deltaTime); - This will set the transform movement to rotate and apply the speed variable to the rotates transform movement.

}

}

1. Save the code and go back to Unity.
2. Drag the code file onto the cube – This will make the code run on the cube specifically.
3. Press play to test if the code works.

<https://www.youtube.com/watch?v=TXFMmqdfn54>