Functions and If statements

Objective

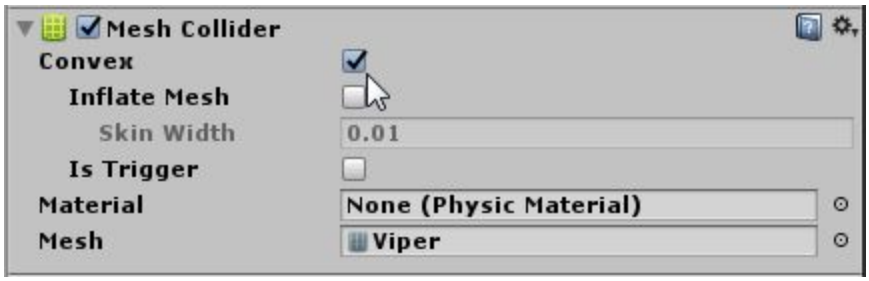
To show logic flow through a program

The if statement

(re)Introducing the **If statement**

Setup

- Load the provided SpaceShip package.
- Place the spaceship into the scene and place a Rigidbody onto the ship
- Turn off the gravity on the Rigidbody.
- Add a collider to the mesh and set it to **Convex**



- Create a new script called **ShipController** and attach it to the ship
- Place the camera behind the ship and parent it to the ship GameObject.

Lesson

Open **ShipController** and add this code.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ShipController : MonoBehaviour {

    public float speed = 20;
    public float rollWeight = 2;
    public float pitchWeight = 2;

    void Update()
    {
        float roll = -rollWeight * Input.GetAxis("Horizontal");
        float pitch = pitchWeight * Input.GetAxis("Vertical");
        Vector3 Rotation = new Vector3(pitch, 0, roll);

        transform.Rotate(Rotation);

        transform.position += transform.forward * speed * Time.deltaTime;
    }

}
```

This is all you need for a super simple controller, Press play and fly around.

Now let’s let the player control the speed

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ShipController : MonoBehaviour {

    public float speed = 20;
    public float rollWeight = 2;
    public float pitchWeight = 2;
    public float thrust = 5;

    void Update()
    {
        float roll = -rollWeight * Input.GetAxis("Horizontal");
        float pitch = pitchWeight * Input.GetAxis("Vertical");
        Vector3 Rotation = new Vector3(pitch, 0, roll);

        transform.Rotate(Rotation);

        if (Input.GetButton("Jump"))
        {
            speed = speed + thrust;
        }

        transform.position += transform.forward * speed * Time.deltaTime;
    }

}
```

Play the game now and see what happens.

Still a little broken, but let's take a bit of a detour.

We have used If Statements before, but we want to go over them again so you know exactly how they work

```
if (This condition is true)) {
      Then do this line of code
}
```

What we need to do is introduce a new part of this if statement, the **else.**

```
if (This condition is true)) {
      Then do this line of code
}else (if the condition is not true){
      Do this line of code
}
```

Let’s use this to fix one of our issues.

First change the If statement to this If / Else

```
        transform.Rotate(Rotation);

        if (Input.GetButton("Jump"))
        {
            speed = speed + thrust;
        }
        else
        {
            speed = speed - thrust;
        }

        transform.position += transform.forward * speed * Time.deltaTime;
    }

}
```

Now when we don’t press space the ship will slow down until it starts going backwards.

Add two new variables at the top of the script.

```
public class ShipController : MonoBehaviour {

    public float speed = 20;
    public float rollWeight = 2;
    public float pitchWeight = 2;
    public float thrust = 5;

    public float maxThrust = 100;
    public float minThrust = 10;

    void Update()
    {
        float roll = -rollWeight * Input.GetAxis("Horizontal");
        float pitch = pitchWeight * Input.GetAxis("Vertical");
        Vector3 Rotation = new Vector3(pitch, 0, roll);

        transform.Rotate(Rotation);

        if (Input.GetButton("Jump"))
        {
            speed = speed + thrust;

            if (speed > maxThrust)
            {
                speed = maxThrust;
            }

        }
        else
        {
            speed = speed - thrust;

            if (speed < minThrust)
            {
                speed = minThrust;
            }
        }

        transform.position += transform.forward * speed * Time.deltaTime;
    }

}
```

This should now fix our script

Outcome

You should have a ship that can fly around and also has a basic understanding of if statements.

More If's

Adding more if's to the game

Setup

Continued from the last lesson.

Lesson

Let's make the player respawn if they go out of bounds.

```
public float pitchWeight = 2;
public float thrust = 5;

public float maxThrust = 100;
public float minThrust = 10;

//Stored info on where the player starts
public Quaternion startingRotation;
public Vector3 startingPosition;


void Start()
{
    startingRotation = transform.rotation;
    startingPosition = transform.position;
}


void Update()
{

    //If the player flys too low they will respawn;
    if (transform.position.y < -50)
    {
        transform.position = startingPosition;
        transform.rotation = startingRotation;
    }

    float roll = -rollWeight * Input.GetAxis("Horiz
    float pitch = pitchWeight * Input.GetAxis("Vert
    Vector3 Rotation = new Vector3(pitch, 0, roll);
```

Play the game now and try and fly very low and you should see your ship reset.

Let's modify this line a little

```
//If the player flys too low they will respawn;
if (transform.position.y < -50 || transform.position.y > 300)
{
    transform.position = startingPosition;
    transform.rotation = startingRotation;
}
```

Now play the game and try and fly up.

What is happening here is we are asking:

```
if (This condition is true (or) If this condition is true))
{
      Then do this line of code
}
```

So it works if the player is less than Y -50 **OR** if the player is above y 300.

Try writing more lines of code for flying too far left, right, forwards and backwards.

```
void Update()
{

    //If the player flys too low they will respawn;
    if (transform.position.y < -50 || transform.position.y > 300)
    {
        transform.position = startingPosition;
        transform.rotation = startingRotation;
    }

    if (transform.position.x < -300 || transform.position.x > 300)
    {
        transform.position = startingPosition;
        transform.rotation = startingRotation;
    }

    if (transform.position.z < -300 || transform.position.z > 300)
    {
        transform.position = startingPosition;
        transform.rotation = startingRotation;
    }


    float roll = -rollWeight * Input.GetAxis("Horizontal");
```

Run and test to see if this works.

Also you can then add in this function to reset the player once they hit an object.

```
void Start()
{
    startingRotation = transform.rotation;
    startingPosition = transform.position;
}

void OnCollisionEnter()
{
    transform.position = startingPosition;
    transform.rotation = startingRotation;
}

void Update()
{
```

Notice now we are repeating ourselves 4 times, we should fix this.

Remember D.R.Y. (Don't repeat yourself)

In coding if you have written the same commands many times over you are most likely doing it wrong.

So let's move all these reset commands into a new function called ResetPlayer

```
void Start()
{
    startingRotation = transform.rotation;
    startingPosition = transform.position;
}

void ResetPlayer()
{
    transform.position = startingPosition;
    transform.rotation = startingRotation;
    GetComponent<Rigidbody>().angularVelocity = Vector3.zero;
    GetComponent<Rigidbody>().velocity = Vector3.zero;
}

void OnCollisionEnter()
{
    transform.position = startingPosition;
    transform.rotation = startingRotation;
}

void Update()
```

Now let's call ResetPlayer everytime we want to reset the player

```
void ResetPlayer()
{
    transform.position = startingPosition;
    transform.rotation = startingRotation;
    GetComponent<Rigidbody>().angularVelocity = Vector3.zero;
    GetComponent<Rigidbody>().velocity = Vector3.zero;
}

void OnCollisionEnter()
{
    ResetPlayer();
}

void Update()
{

    //If the player flys too low they will respawn;
    if (transform.position.y < -50 || transform.position.y > 300)
    {
        ResetPlayer();
    }

    if (transform.position.x < -300 || transform.position.x > 300)
    {
        ResetPlayer();
    }

    if (transform.position.z < -300 || transform.position.z > 300)
    {
        ResetPlayer();
    }


    float roll = -rollWeight * Input.GetAxis("Horizontal");
    float pitch = pitchWeight * Input.GetAxis("Vertical");
    Vector3 Rotation = new Vector3(pitch, 0, roll);
```

For more information about functions, view the Functions handout.

And as you do this you can clean up the code more and make a new function:

Outcome

The students should have a basic understanding of how functions work.

Further tasks for students to work through

- Add in the shooting from the tank game
- Make objects that can be destroyed
- Terrain, Probuilder, and Animations make a fun level to fly around.