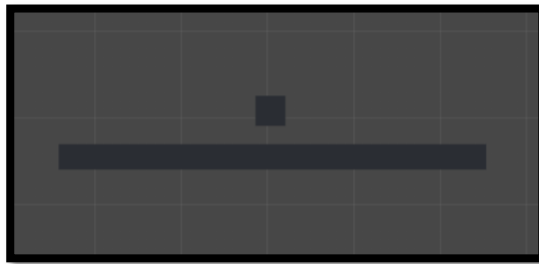


Programming tutorial – 2D character movement and jumping in Unity

How to get a character to move in a 2D project in Unity:

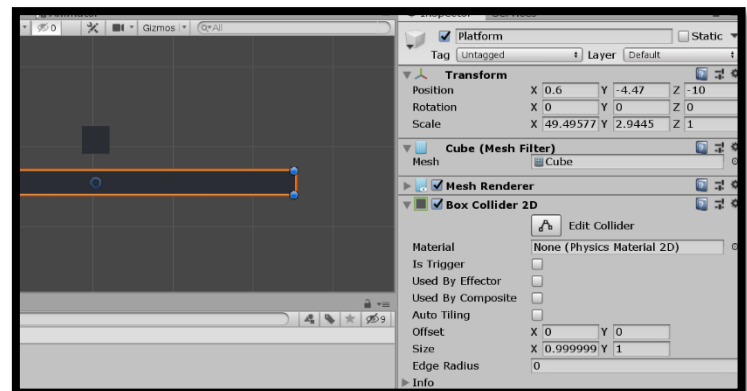
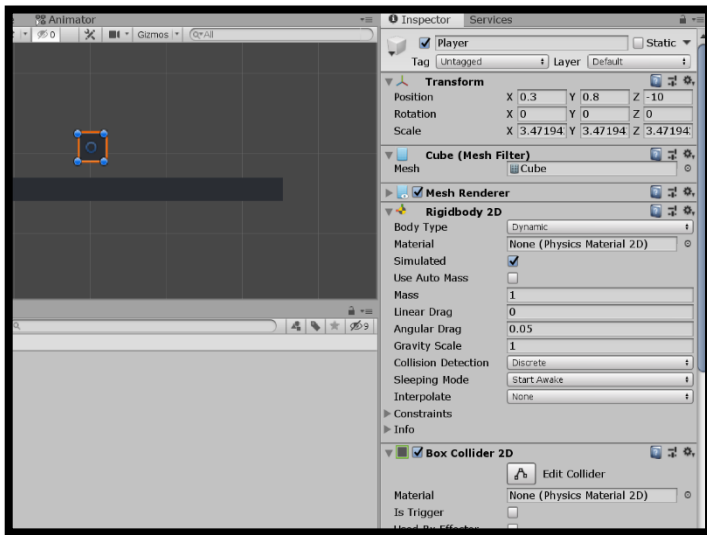
1. Set up the scene

Right click in the Hierarchy section of the project and create a 3D cube or sphere, this will be your character's body, add a collider (2D) and rigidbody (2D) to this and name it "Player". Create another cube and elongate it to create the platform beneath the player, add a collider (2D) to this as well but do not add a Rigidbody, name this object "Platform". The scene should look like this:



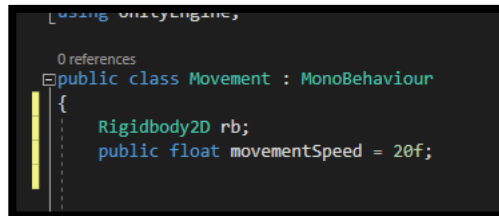
(The 'Player' gameobject)

(The 'Platform' gameobject)



2. Moving left and right

Add a script called 'Movement' to the player object and open it. Make a reference to the rigidbody of the object by typing "rigidbody2D rb;" under the public class function. Below this, type "public float movementSpeed;" which is a public variable that will hold the speed of the character. Type "= 20f;" next to this to set a default speed of 20. You can adjust this anytime in the inspector in the unity project. It should look like this:

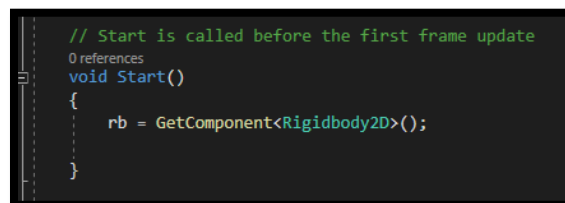


```
using UnityEngine;

0 references
public class Movement : MonoBehaviour
{
    RigidBody2D rb;
    public float movementSpeed = 20f;
}
```

Once the above is done, find the rigidbody of the object by typing:

"rb = GetComponent<Rigidbody2D>();" into the Start() function.



```
// Start is called before the first frame update
0 references
void Start()
{
    rb = GetComponent<Rigidbody2D>();
}
```

This gets the object's rigidbody component.

In the Update() function, create a 'Move' variable which checks which arrow key you press to go horizontally:

"float Move = Input.GetAxis("Horizontal");"

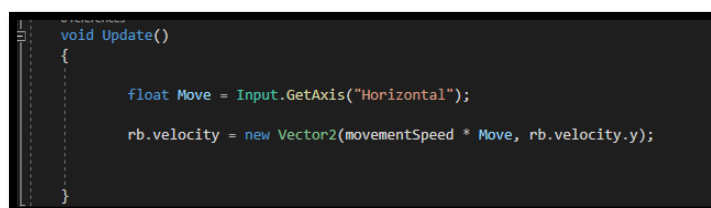
The reason as to why the move variable is a float is because in the script, the 'GetAxis' function works with variables that have values of either -1 or 1 (or 0) so left would be -1 and right would be 1.

Beneath this, write the final line of code to get the player character to move, it is:

"rb.velocity = new Vector2(movementSpeed * Move, rb.velocity.y);"

This line of code states that the velocity of the rigidbody of the object is equal to the movement speed multiplied by the -1/1 value in the horizontal axis. If the velocity is negative, the object goes left, and if it is positive, the object goes right. 'Rb.velocity.y' means there is no change in the Y axis.

These two lines of codes should look like this in the script:



```
0 references
void Update()
{
    float Move = Input.GetAxis("Horizontal");
    rb.velocity = new Vector2(movementSpeed * Move, rb.velocity.y);
}
```

Save this and go back to the Unity project and press play, you should be able to make the player move left and right using the left and right arrow keys.

3. Jumping

If you want to make your player jump as well as move, go back to the script and below the code in the Update() function, type:

```
if(Input.GetKeyDown(KeyCode.Space))
{
    rb.AddForce(new Vector2(0,10), ForceMode2D.Impulse);
}
```

This code checks whether the player has pressed the space key and if so, adds an upwards force to the player object which makes it jump.

If you want to be able to alter the upward force within the inspector, add a new public float called “public float jumpSpeed = 10f;”:

```
public float jumpSpeed = 10f;
```

and change the script within the jump code to:

```
rb.AddForce(new Vector2(0,jumpSpeed), ForceMode2D.Impulse);”.
```

This should look like this in the script:

```
if(Input.GetKeyDown(KeyCode.Space))
{
    rb.AddForce(new Vector2(0, jumpSpeed), ForceMode2D.Impulse);
}
```

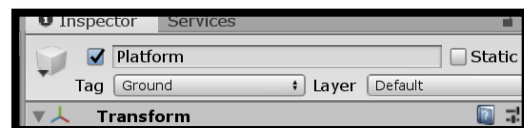
4. Checking if the player is ‘grounded’

At the moment, when you press play in the unity project and press the space bar multiple times, the player character jumps infinitely – even without being near the ground. To stop this from happening, you will need to do an ‘isGrounded’ check within the script.

Go back to the ‘Movement’ script and create a new boolean variable called “public bool isGrounded;” and add a tag to the ‘Platform’ gameobject called “Ground”. These should both look like this:

```
public bool isGrounded;
```

(the ‘isGrounded’ Boolean)



(‘Platform’ tagged with “Ground”)

Also make a reference to the ‘Platform’ object by adding a ‘public GameObject Platform;” variable and find its collider by typing “Platform.GetComponent<Collider>();” into the Start() function. Save, go back to the Unity project, and drag the platform gameobject into the new variable box in the inspector.

This should look like:

```
public GameObject Platform;

// Start is called before the first frame update
0 references
void Start()
{
    Platform.GetComponent<Collider>();
}
```

Below the Update() function (not in it), add an OnCollisionEnter2D function, (or just OnCollisionEnter if the game is 3D):

```
private void OnCollisionEnter2D (Collision2D collision)
```

```
{
```

```
}
```

```
“
```

And within it write an if statement to check whether the object collides with another object with a certain tag or not (in this case the platform). If it does, 'isGrounded' is true. The if statement should look like:

```
if(collision.gameObject.tag == "Ground")
```

```
{
```

```
isGrounded = true;
```

```
}”
```

Then add an OnCollisionExit2D function under that:

```
private void OnCollisionExit2D(Collision2D collision)
```

```
{
```

```
}
```

```
“
```

And within it write the same as above, but with 'isGrounded' being false:

```
if(collision.gameObject.tag == "Ground")
```

```
{
```

```
isGrounded = false;
```

```
}”
```

This should look like:

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Ground")
    {
        isGrounded = true;
    }
}

0 references
private void OnCollisionExit2D(Collision2D collision)
{
    if (collision.gameObject.tag == "Ground")
    {
        isGrounded = false;
    }
}
```

This will make it so when you enter the collider (touch the ground), isGrounded is true. When you exit the collider (leave the ground), isGrounded is false.

After this, add “&& isGrounded == true” to the end of the “if(Input.GetKeyDown(KeyCode.Space))” script line in the Update() function of the script. It should look like:

“if(Input.GetKeyDown(KeyCode.Space) && isGrounded == true)”

This will ensure that you can only jump if you press space whilst on the ground. If you press space whilst in the air, the code within the brackets will not work.

In the script, this should look like:

```
if(Input.GetKeyDown(KeyCode.Space) && isGrounded == true)
{
    rb.AddForce(new Vector2(0, jumpSpeed), ForceMode2D.Impulse);
}
```

The entire script should look like:

```
public class Movement : MonoBehaviour
{
    public float movementSpeed = 20f;

    Rigidbody2D rb;
    public float jumpSpeed = 10f;

    public bool isGrounded;

    public GameObject Platform;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        Platform.GetComponent<Collider>();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        float Move = Input.GetAxis("Horizontal");

        rb.velocity = new Vector2(movementSpeed * Move, rb.velocity.y);

        if(Input.GetKeyDown(KeyCode.Space) && isGrounded == true)
        {
            rb.AddForce(new Vector2(0, jumpSpeed), ForceMode2D.Impulse);
        }
    }

    0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.tag == "Ground")
        {
            isGrounded = true;
        }
    }

    0 references
    private void OnCollisionExit2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Ground")
        {
            isGrounded = false;
        }
    }
}
```