Make your own 2D Platformer

# Introduction

This tutorial will take you through the steps required to develop a simple 2D platformer game using Unity and C#.

You will be able to design your own levels using Unity, which you can then give to friends and family to play.

# Required Software

Unity is the only required piece of software, make sure MonoDevelop is installed when installing Unity.

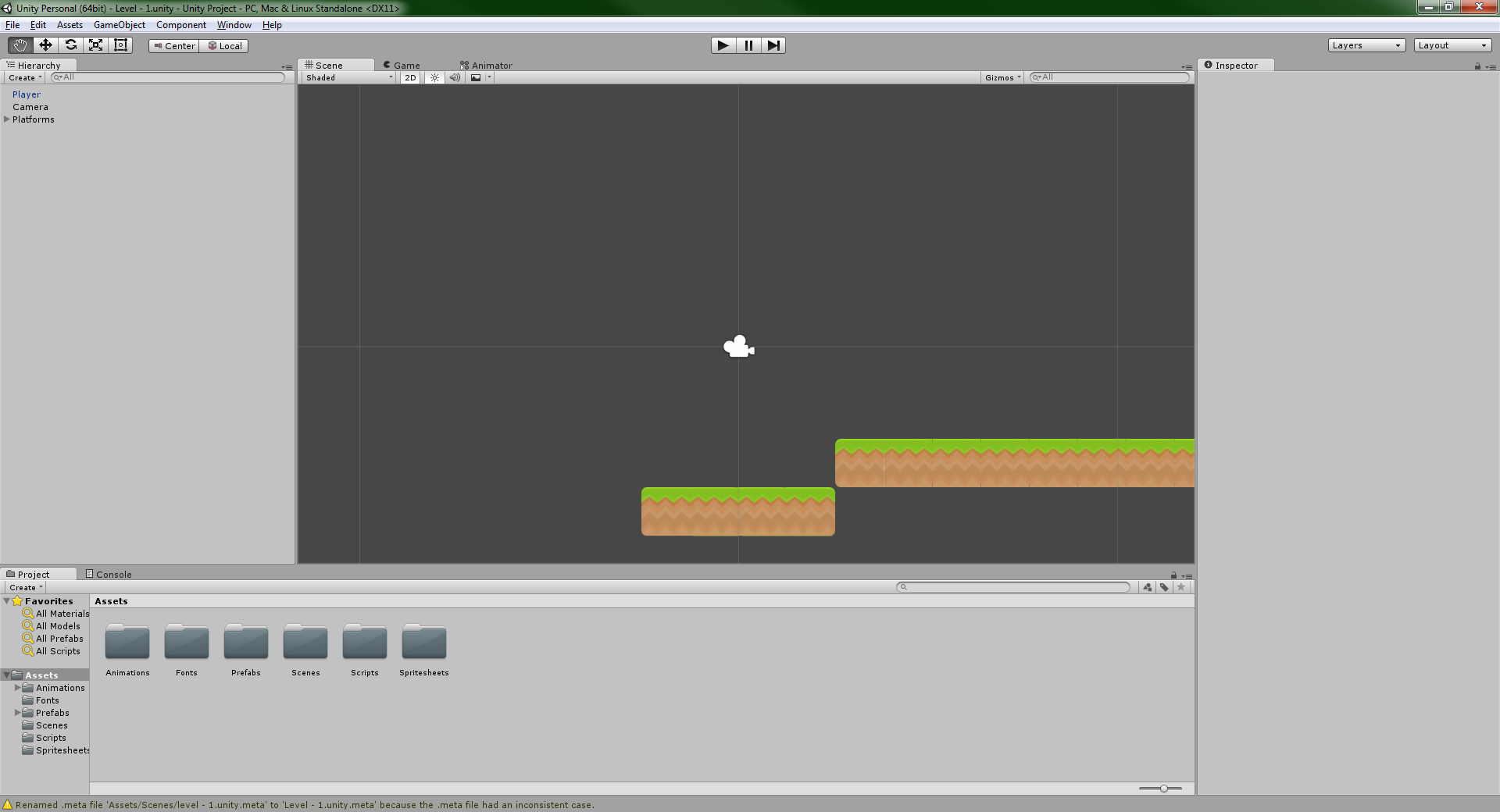
<http://www.unity3d.com/get-unity>

# Let’s Begin

## Starting

* Navigate to the Unity Project folder
* Open the Assets folder, then the Scenes folder
* Open the file called ‘Level - 1’

## Unity Interface



When you open the Level – 1 file you will be presented with this. There is a play button at the top-middle of the screen, give that a press and you will see the beginnings of your game.

At the top left of the screen there is a window called Hierarchy, this is a list of everything that is in your game at the moment.

To the Right of the Hierarchy window is the Scene window, this is where everything that in in your game is located. You can use the arrow keys to move around to see where everything is.

The Game tab that is next to the Scene tab is what you will see when you first press play.

On the right hand side of the screen is the Inspector window, this is where you can see the details of a particular game object.

Finally, the Project window at the bottom of the screen is where all of your files are kept.

## Getting Started

You may have noticed when you pressed play that the Green Space Creature just fell off the bottom of the screen. In order to fix that we are going to modify a variable. A variable is word that is connected to a number. We want to change how far left or right the player starts.

In the Hierarchy window click on the Player, now you will have a whole bunch of information show up in the Inspector window. Don’t worry, we won’t worry about most of it for now. At the top of the Inspector window is a section called Transform, and just under that is the word Position. The X next to Position is how far left or right the player is. Let’s change it so it’s 0. Press play again and the player will now land in the middle of the platform.

## Movement

Yay! The player no longer falls off of the bottom of the screen, but it can’t do anything. Now we’ll make it move. In the project window navigate to the Scripts window and open the Player Script. Add the following code just below the ‘Player Update’.

|  |  |
| --- | --- |
| //--------------------------------  // Player Update   |  | | --- | | movementAmount = Input.GetAxis ("Horizontal");  animator.SetFloat("Speed", Mathf.Abs(movementAmount)); | |

And now the following to the FixedUpdate function. (Hint: in between the ‘{‘ and ‘}’ that come after ‘void FixedUpdate()’.

|  |  |
| --- | --- |
| void FixedUpdate()   |  | | --- | | if (movementAmount != 0.0f && Mathf.Abs (rigidBody.velocity.x) < 3) {  rigidBody.AddForce (new Vector2 (movementAmount \* 3, 0.0f));  } |   {  } |

Save the file and then go to back to Unity and press the Play button. Now the player can move left and right, hurrah! The player does move very slowly though, let’s make a variable! Add this code to the top right below the moveAmount.

|  |  |
| --- | --- |
| |  | | --- | | [SerializeField] float movementSpeed = 1; |   float movementAmount; |

And change the code in the in the FixedUpdate function.

|  |  |
| --- | --- |
| void FixedUpdate()  {  }   |  | | --- | | if (movementAmount != 0.0f) {  rigidBody.AddForce (new Vector2 (movementAmount \* movementSpeed, 0.0f));  } | |

Save the file and go back into Unity. Click on the Player in the Hierarchy then look at the Inspector window, scroll down to where the Player script is, you will see that Movement Speed is set 1, let’s change it to 10. Hit play.

## Jumping

It would be nice if the player could get up onto the platform that’s to the right. To do that we would need to be able to jump. Add the following code into the Update function below the code we added earlier.

|  |  |
| --- | --- |
| //--------------------------------  // Player Update  movementAmount = Input.GetAxis ("Horizontal");  animator.SetFloat("Speed", Mathf.Abs(movementAmount));   |  | | --- | | jumpAmount = Input.GetAxis ("Jump");  if (jumpAmount != 0.0f) {  animator.SetBool ("Jump", true);  } else {  animator.SetBool ("Jump", false);  } | |

And add this just below the movementSpeed variable that you just added.

|  |  |
| --- | --- |
| float movementAmount;   |  | | --- | | float jumpAmount; | |

Now add the following to the FixedUpdate function.

|  |  |
| --- | --- |
| void FixedUpdate()  {  if (movementAmount != 0.0f) {  rigidBody.AddForce (new Vector2 (movementAmount \* movementSpeed, 0.0f));  }   |  | | --- | | if(jumpAmount != 0.0f){  rigidBody.AddForce (new Vector2 (0.0f, jumpAmount \* 40));  } |   } |

Save and go back into Unity and play. Well the Player can now jump, but can also jump forever, that’s a bit of a problem, and we can fix that by checking if the Player is on the ground before make the Player jump. Add the following variables below the jumpAmount variable that we added earlier.

|  |  |
| --- | --- |
| float jumpAmount;   |  | | --- | | float groundRadius = 0.1f;  bool isGrounded = false;  [SerializeField] LayerMask whatIsGround; | |

Jump back into Unity and look at the Player in the Inspector as we did before and you will see a new variable has been added, click the drop down box next to What Is Ground and select ‘Ground’.

Next add the following to the Update function.

|  |  |
| --- | --- |
| void Update()  {  movementAmount = Input.GetAxis ("Horizontal");  animator.SetFloat("Speed", Mathf.Abs(movementAmount));  jumpAmount = Input.GetAxis ("Jump");   |  | | --- | | Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, groundRadius, whatIsGround);  isGrounded = false;  for (int i = 0; i < colliders.Length; i++) {  if (colliders [i].gameObject != gameObject) {  isGrounded = true;  }  } |   if (jumpAmount != 0.0f) {  animator.SetBool ("Jump", true);  } else {  animator.SetBool ("Jump", false);  }  } |

Now change the section of code that makes the Player jump.

|  |  |
| --- | --- |
| void FixedUpdate()  {  if (movementAmount != 0.0f) {  rigidBody.AddForce (new Vector2 (movementAmount \* movementSpeed, 0.0f));  }  Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, groundRadius, whatIsGround);  for (int i = 0; i < colliders.Length; i++) {  if (colliders [i].gameObject != gameObject) {  isGrounded = true;  }  }  rigidBody.AddForce (new Vector2 (0.0f, jumpAmount \* 40));   |  | | --- | | if(jumpAmount != 0.0f && isGrounded == true){ |   }  } |

Save and go back to Unity and play! It works but something doesn’t look quite right. The Player only changes to his jumping animation when we hold down the jump key. It would be better if the animation was always playing while he was jumping.

|  |  |
| --- | --- |
| void Update ()  {  //--------------------------------  // Player Update  movementAmount = Input.GetAxis ("Horizontal");  animator.SetFloat("Speed", Mathf.Abs(movementAmount));  jumpAmount = Input.GetAxis ("Jump");   |  | | --- | | if (isGrounded == false) { |   animator.SetBool ("Jump", true);  } else {  animator.SetBool ("Jump", false);  }  // End Player Update  //--------------------------------  } |