Contents

[Unity2D and Sprites 1](#_Toc9859801)

[Setting the Stage/Scene 2](#_Toc9859802)

[Exploring what’s different 3](#_Toc9859803)

[Sprites 4](#_Toc9859804)

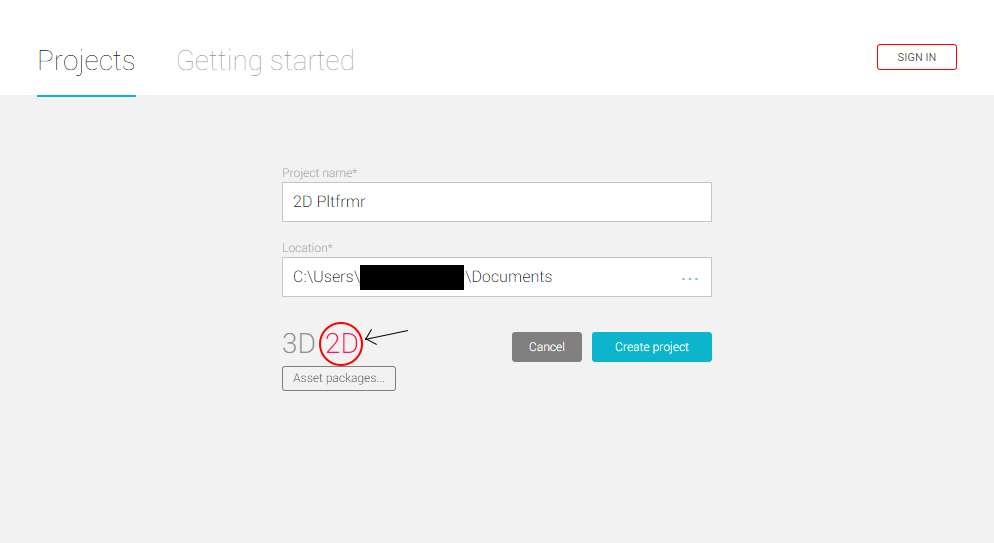
[Adding Sprites to your Game 10](#_Toc9859805)

[Placing Sprites into a Level 12](#_Toc9859806)

# Unity2D and Sprites

fdsfadfs

# Setting the Stage/Scene

The first step, as with any Unity project, is to create the project and your first scene.

This is where developing a 2D game in Unity begins to differ from 3D game projects in Unity. We need to make slight changes to how we setup out project.

1. Create a new Unity Project.
2. Give your project a descriptive name, such as ‘2D Testing.
3. Change the type of project from ‘3D’ to ‘2D’ by clicking on the ‘2D’ label that’s on-screen.

# Exploring what’s different

With our project and scene set up, we can begin to see what is different.

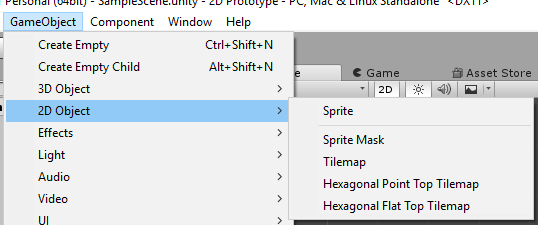
## The View

The scene view has now lost all perspective. All objects are placed and moved around as if placing paper cut-outs onto a table.

## The Scene

The scene that was created for us doesn’t have a light in it, like a 3D project would. 2D games can often do without any lighting, so they are left out by default. You can still add lights if you’d like.

# Sprites

You can continue creating 3D game objects in your 2D game and that will continue to work fine. However, Unity provides 2D objects that you can create for your 2D games

These are called Sprites

A Sprite is a 2D image with some extra properties to set it apart from any other image.

There are other object that we will look at in the tilemap tutorial

To Create a Sprite

1. Select GameObject -> 2D Object
   1. Select Sprite

This will create a Sprite game object into the game

## Distinguishing Features

When you create a sprite, a typical Game Object is created. It has the usual Transform component, for position, rotating and scaling your object.

It has one new component we haven’t seen before.

### Sprite Renderer

As the name suggests, this component is for displaying a sprite on the screen. There are some handy features to be found!

#### Sprite

This is where you add your sprite to be displayed

#### Color

We can add a colour tint to our image. Perhaps we can have a flash of red on our character when they take damage.

#### Flip

Rather than drawing a bunch more sprites for our characters facing left, right, up and down, the Sprite Renderer allows us to flip our sprites in any of those 4 directions.

#### 

#### Sorting Layer and Order

Since everything is flat in 2D, we need to decide what sprites should appear in front of others. We can use Sorting Layers to group sprites together, such as having the background, middle ground and background sprites all together. Within each layer, we can use Order in Layer to bring different objects to the front, such as our main character or a door on a building.

#### Mask interaction

A mask is similar to the mask used in image editing. A mask, hides part of an image while showing the rest. It uses a grey scale image to choose what is seen and what is not.

## Creating a Sprite Asset

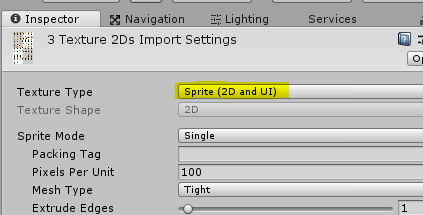
For our sprite gameObject to have an image, we need a sprite asset. You can import an image into Unity.

1. Import the Images in the resource file provided

You should have 3 different images in your Assets folder

When we are not in a 2D project it will come up as being a “Texture”, which is essentially just a plain old image.

For our sprites, we need it in a 2D Sprite texture type, which should automatically be set up when in a 2D project mode.

However, if it does not (Or we set it up from a 3D game mode). We can select the image in our assets folder, and change the Texture Type

1. Select the assets

* In the Inspector

1. Make sure the Texture Type is set to Sprite(2D and UI)
2. Hit Apply to save

The should now be set to Sprites. The transparent backgrounds should seem to work better now.

# Adding your sprite

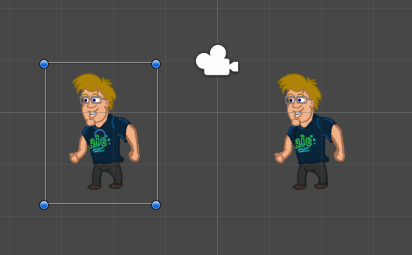
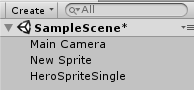
Now that we have our sprites, let’s add one to the scene

There are two ways you can do this

* With the created Sprite

1. Pass in the HeroSpriteSinge into the sprite variable

Or

1. Click and drag the sprite asset into the scene

Both should create a sprite of the character into the scene.

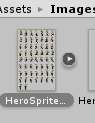
However, if we were to bring in the other sprites. You would bring in all of the images the sprite sheet contains. We need a way to split up the sprite sheet

# Image Sprite Import Settings

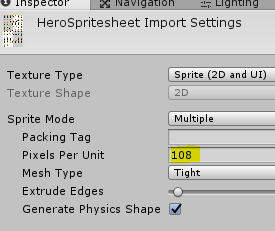
When we select our image asset, in the inspector there are a few settings we can edit to

### Sprite Mode

Sprite mode, determines the type of sprite it is.

If you happen to have a bunch of smaller images packed together, you can switch this options from ‘Single’ to ‘Multiple’, then slice it up using the Sprite Editor.

Multiple sprites packed together are known as a ‘Sprite Sheet’.

1. Select the HeroSprites image in the Assets folder.
2. Change its Sprite Mode from Single to Multiple and click Apply.

### Pixels Per Unit

This option allows us to set what the default size fits in the Unity Unit (a square in the graph in the scene)

The pixel size of our character on the x (side) is 108 px

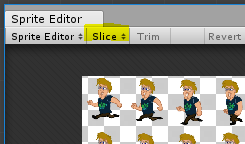
1. Set the Pixels Per Unit to 64

Now we can split our sprites in the Sprite editor

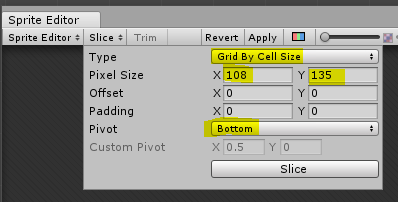
## Sprite Editor

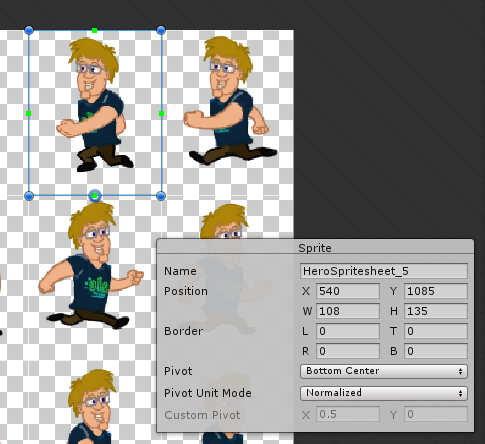
The sprite editor is a tool for editing sprites!

You can crop your sprites, choose the point that your sprite will rotate around and slice them up into smaller images. This can be done manually, and you can have different sizes. However they all need to be squares.

In many cases, Unity can be very clever and split up our image for us. As it did in this case!

If you have a character that you’ll be controlling in your game, you may want to move the Pivot down to your character’s feet (or equivalent). Moving characters around is much easier when their pivot is at the feet.

1. With the HeroSprites image selected, click the Sprite Editor button.
2. Click on the Slice button in the top-left corner of the window.
   1. Open the Pivot dropdown and click Bottom.
   2. Change type to Grid by Cell Size
   3. Size for the image is
      1. X: 108
      2. Y: 135
   4. Click the Slice button at the bottom of the box.
3. Click Apply in the top-right.
4. Close the Sprite Editor.

You can now see a series of white boxes around each character. They may look large, but they fit all versions of the character, allowing us to keep a consistent size for each assessment.

You can select the box and make adjustments.to each element.

A sprite info box will appear where you can make changes to the object, You can change all information about the selection, even name the internal sprite.

The Pivot point is a hollow blue circle

If you now go to the Assets folder, you will see an arrow next the the sprite. If you click it, you can see all the split up sprites within it.

You can use these sprites to individually bring into the scene.

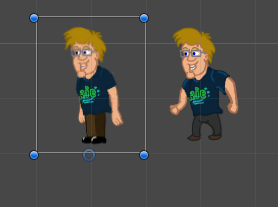


# Fixing the Size

If we compare the size of the individual sprite to the sprite sheet version, you will see that they are a little different in size.

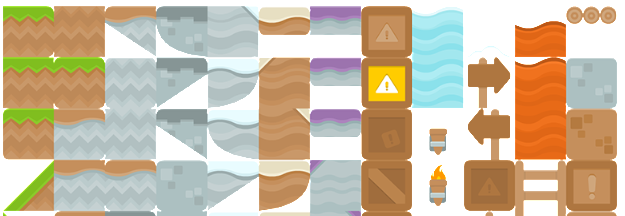
That is because the sprite sheet characters are double the size of the individual sprite, to accommodate for the legs and arm animations.

Because of this, we need to half 108 to 54

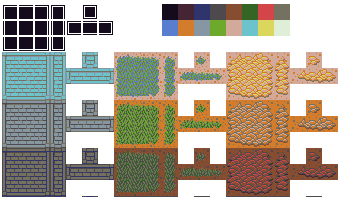
This will make them two units big but they will be the same size. The size of the sprite will not affect our physics.

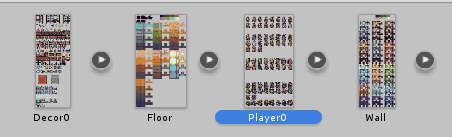
# Tile Sprite map

Just like with our character, our tilesheets need to be cut up too.

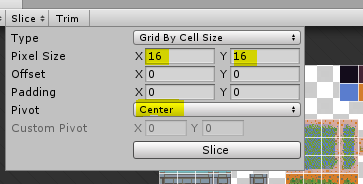
Depending on the spritesheet, will depend on the box size.



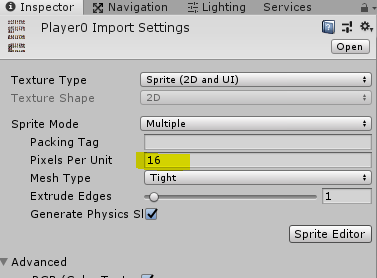
Platformer Tileset provided is 64 by 64 pixels a tile



The top down dungeon Tileset images are 16 by 16

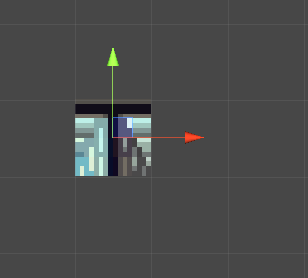
1. Cut up the tileset so each tile is individual
   1. Set up Sprite Mode to Multiple
   2. Open up Sprite Editor
   3. Select Slice
   4. Set Pixel Size
   5. Keep pivot to center

We keep pivot to Center as the drawing of the tilemap requires it to draw it on the object

1. Set Pixels per unit on the image to be the size of the tile

If there are two size, choose the x size

You should now have tiles that can be brought into the game and fit the Unit size in the Scene



# Fixing the Quality of our Sprites

You will notice that the tiles do not look clean at the edges. They may even be a bit blury. We need to make our sprites look a bit clearer.

## Turn off Anti-Aliased

Anti aliasing is used in videogames to smooth texture borders. You usually want this to be on so that your texture doesn’t look “too pixely” but instead smooth on screen.

For the example on the right, the bottom line is smoothed and blurry. This will make it look ok if we see it in a smaller resolution than this. We’ll notice the pixelated borders if we don’t activate the anti aliasing.

1. Select Edit -> Project Settings -> Quality
2. Set Anti Aliasing to Disabled

## Changing filter and compression

Each image has their own filter and compression.

The filter mode determines how the Texture is filtered when it gets stretched by 3D transformations. This can make the image blury in certain conditions, which is good for 3D objects, but for tiles, not so much

1. Make sure Filter Mode is set to Point(no filter)

The Compression determines how compressed the filesize for the image is, the more compressed, the less detail. While essential for games with lots of textures and materials, We want no compression for our tilemaps to make them as clear as possible

1. Set Compression to None

You could also play around with the Max size and resize algorithm to get into more detail.

The tile set should now look sharper and cleaner. It may still have some bugs, but to fix that you go into positing code, which we won’t look at for now.

Next tutorial we will look at a way to draw our level.