



2D Game Design

Today's Agenda

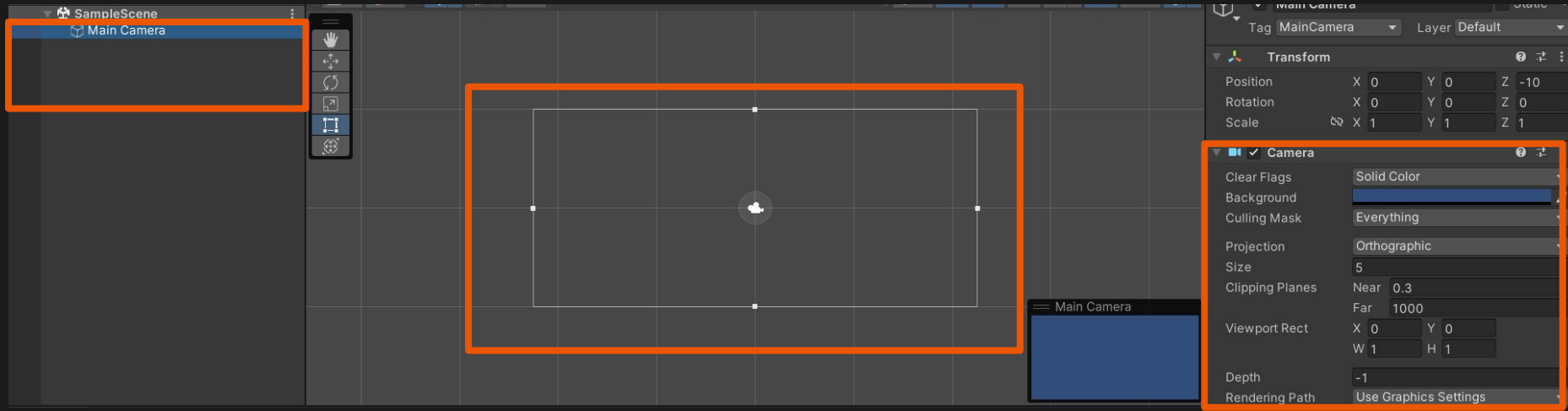


We're going to be going over Camera portion of Chapter 5: Lights and Camera

Chapter 12: 2D Game tools, and Chapter 13: Tilemaps

- Using Orthographic Cameras in a 2D Scene
- Learn about 2D Assets and Tilemaps
- Create Game Objects with 2D Collision and Physics
- Create and Share Your 2D levels

2D Game Project



When we start a 2D Game Project there will be no light source and Camera set into Orthographic Projection. We don't need light because we're not loading any meshes and Orthographic Project is better for 2D Games.

Camera

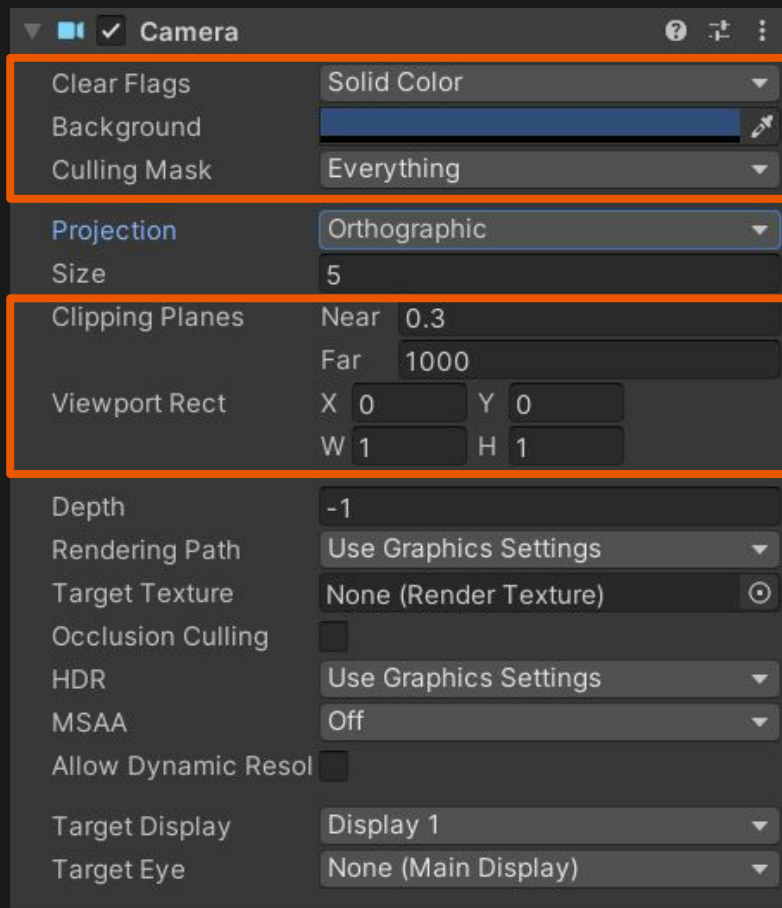


There are two general areas we are [1] controls what's visible, **Clear Flags** describes the background type we have, normally we're used to using **Sky Box** but now we're going to use **Solid Color** defined by the **Background**. While **Culling Mask** defines what the camera should render. Currently is rendering everything.

[2] Controls the size of the camera and it's behaviors. **Clipping Planes** tells us how close or far we need to be for the camera to clip into it, **Viewport** defines how much of the screen this camera should take up, you could have two cameras taking up half the screen with it or so.

1

2

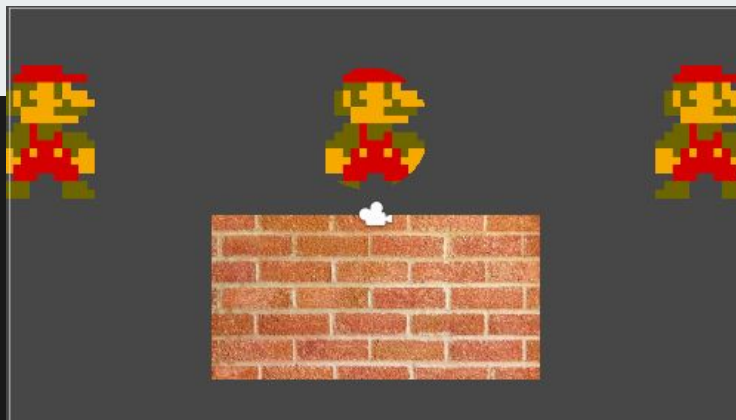


Multiple Cameras



Having multiple **Cameras** with different **Viewports** will allow you to display two different areas of the scene at once.

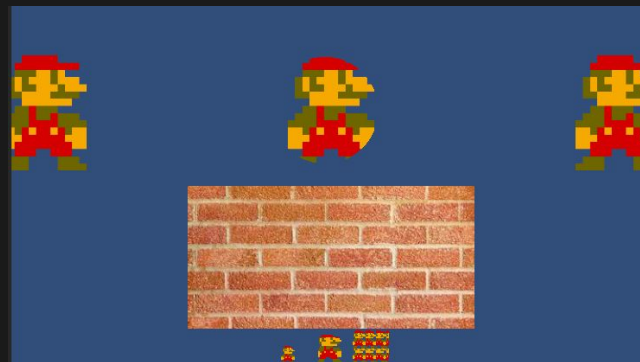
You could show the goals that the player is looking for or split the screen so that multiple players can play at once.



Viewport Rect			
X	0	Y	0
W	1	H	1



Viewport Rect			
X	0.25	Y	-0.2
W	0.5	H	0.3



Perspective vs Orthographic

Perspective Projection is what we've been working with until now, it allows for depth on the 3D axis.

On the other hand Orthographic

Projection removes the 3D axis and makes everything on the same 2D plane.



Sprites

2D Sprite are images that are flat on the screen, they can be anything from pixel art, to hand painted, to picture of real people.

So example of how you can use them, 2D Platformer - Shantae, RogueLite - Binding of Isaac, FPS - Doom Classic, Top Down Farm - Stardew Valley, 2D Fighting - Mortal Kombat, and Point and Click - Paradigm.

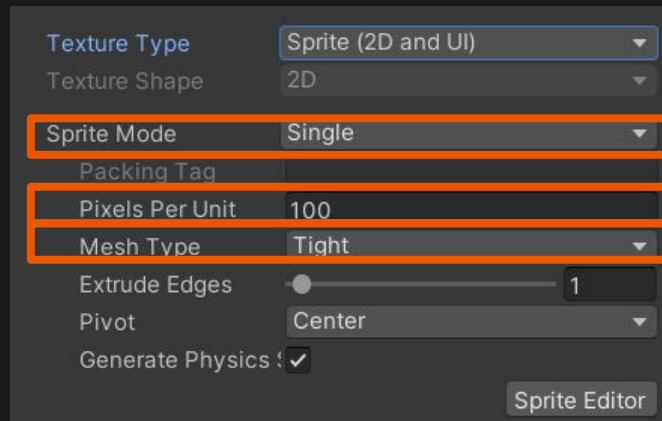


Sprites Import Settings

When working with sprites the import setting on the image has to be set to **Sprite**, unlike when we were working with Textures and it was set to **Default**.

With this we can edit how the image will be used in the Scene.

Sprite Mode allows us to cut up or keep the image as one, **Pixels Per Unit** tells us how much space should the image take in respect to the world space. **Mesh Type** tells us how the Scene should view it as a rectangle the size image width and height or the shape of the Sprite.



Sprites Import Settings



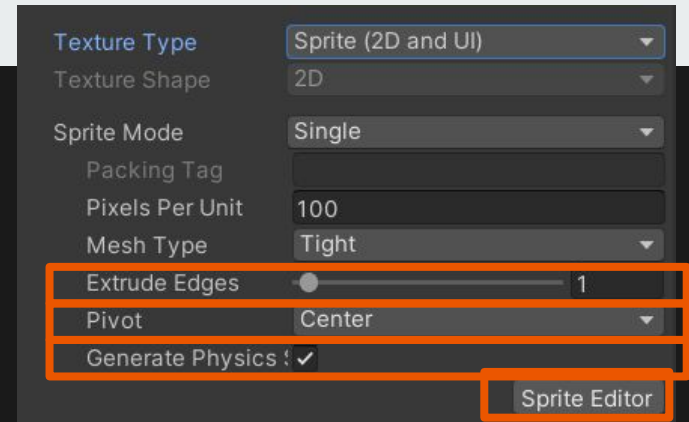
Extruded Edges connect to Mesh type of figure out how much space off of the image should be considered the image.

Pivot is where you want to rotation to occur from and

Generate Physics will build physics based on the Sprites look.

The **Sprite Editor** Button will bring you to the Sprite Editor

View from which you'll be able to edit the 9-Slicing Box that we'll use in a moment.



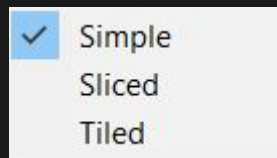
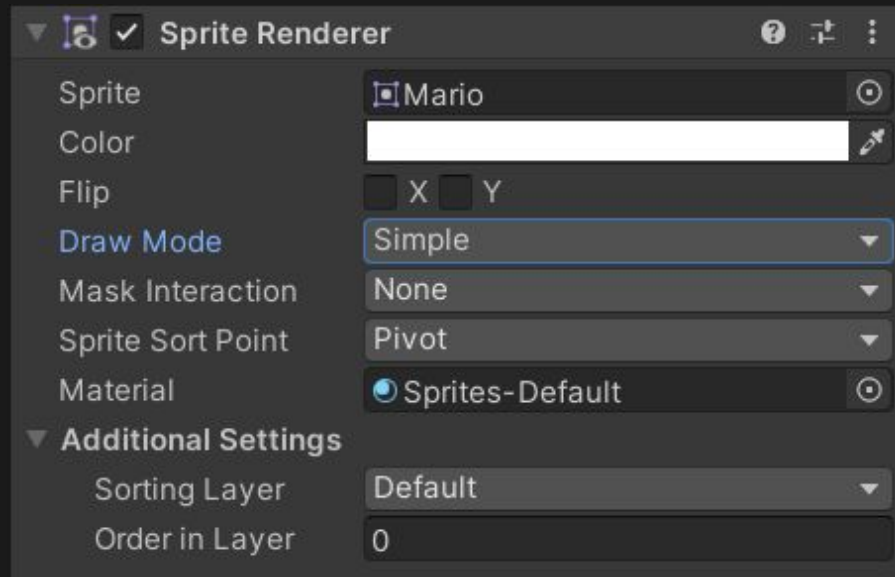
Sprite Renderer



Sprite Renderer is the component through which we can draw the sprite.

The **Sprite** controls what Image is draw, like the mesh choosing the shape of the 3D object.

And **Draw Mode** dictates how it will be drawn and how scaling is dealt with. **Simple** will stretch and compress the image equally.



Sliced And Tiled

For **Sliced** and **Tiled** it's important to know that they are controlled by the 9 Slices box in the Sprite Editor while the Sprite is set for Single Sprite Mode.

Initially the box is filled out so that the center covers all of the image, however you can edit how it's framed such as making the center Mario's body and that being are being affected by the **Slice** and **Tiled** Effects.

Sprite Editor



Challenge: Stretchy Mario

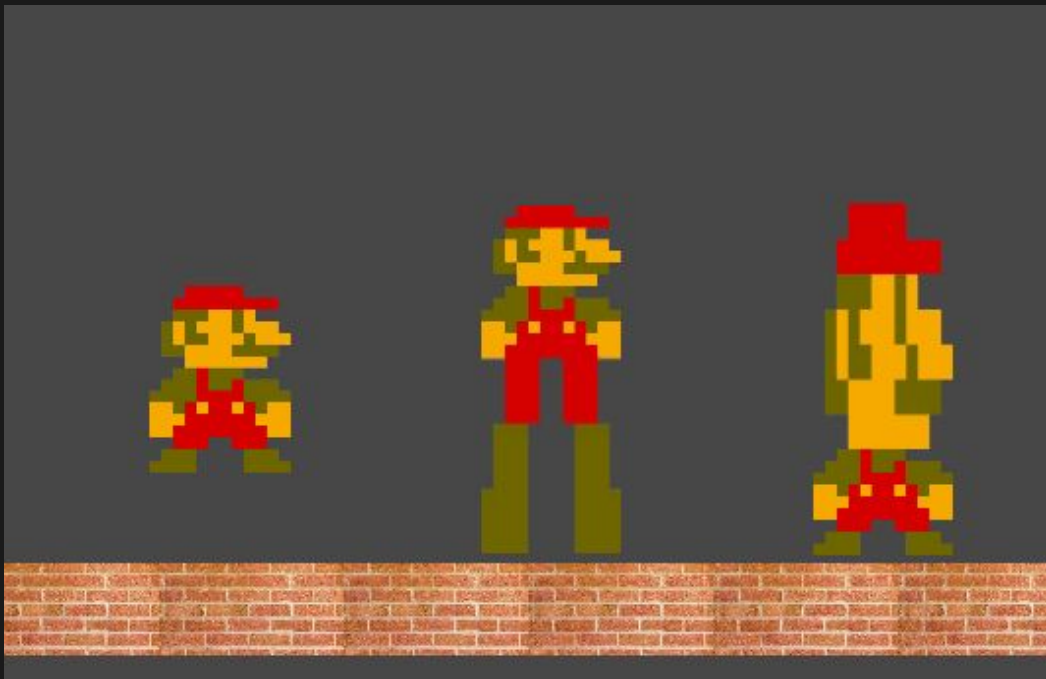


In `Scene_Challenge_0` there are three Marios and a brick wall sprite.

Each Mario has his own Sprite connected to him. Go into the `Sprite Import Settings` and Edit the 9 Points have center over leg and the other over the head.

Then in the Sprite Render for each Mario set them to `Sliced` and stretch the Marios.

For the Brick set it to `Tiled` and build a platform for them to stand on.

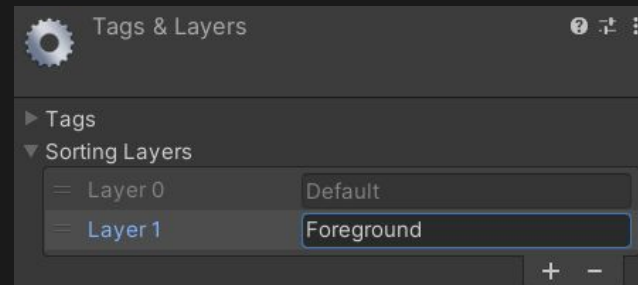
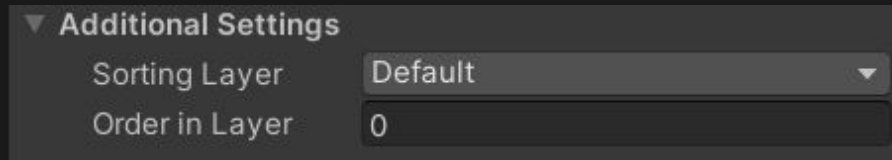


Layers



As there is no **Third Axis** for us to tell which image is further away from the camera the Sprite Renders has the **Sorting Layer** option that tells us which Game Object has priority and should be drawn closer to the camera.

You can even make multiple sorting layers, for items in the foreground, background, or around the player.



Materials - Shaders



As you can see the Sprite Render uses a Material to draw you could swap in one of the **Materials** we've worked on but you will notice that the Sprite is distorted.

That's because the Material is meant for meshes not sprites. To give our sprite different ways of display you need to create a Shader.

Shaders are Scripts that calculator how the image should be drawn, in there you can specify that the material should cling to the Sprite while adding many material effects.

Material

Sprites-Default



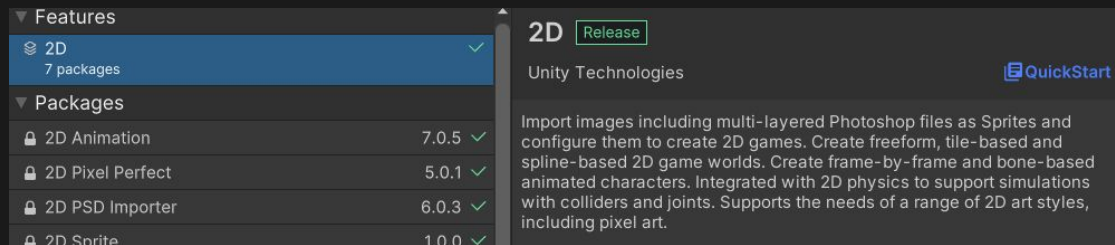
[Get started with 2D Shader Graph in Unity - Dissolve Tutorial](#)

2D Sprites Game Object

When Unity is created in 2D set up it also allows us to create premade 2D Game Objects, these weren't available

to us in 3D because the 2D Package Manager isn't included in the set up.

You can always add this to the 3D projects but searching 2D in the Package Manager.



2D Object	>	Sprites	>	Square
3D Object	>	Physics	>	Circle
Effects	>	Tilemap	>	Capsule
Light	>	Sprite Shape	>	Isometric Diamond
Audio	>	Pixel Perfect Camera		Hexagon Flat-Top
Video	>	Sprite Mask		Hexagon Pointed-Top
UI	>			9-Sliced

Sprite Sheet



Having manage multiple of the same image can be headache. Many animation take several images to give the appearance of a character moving. For that we have **Sprites Sheets**. They're one image that has all of the desired sprites that we can divide up and use as it they we all individual image files.

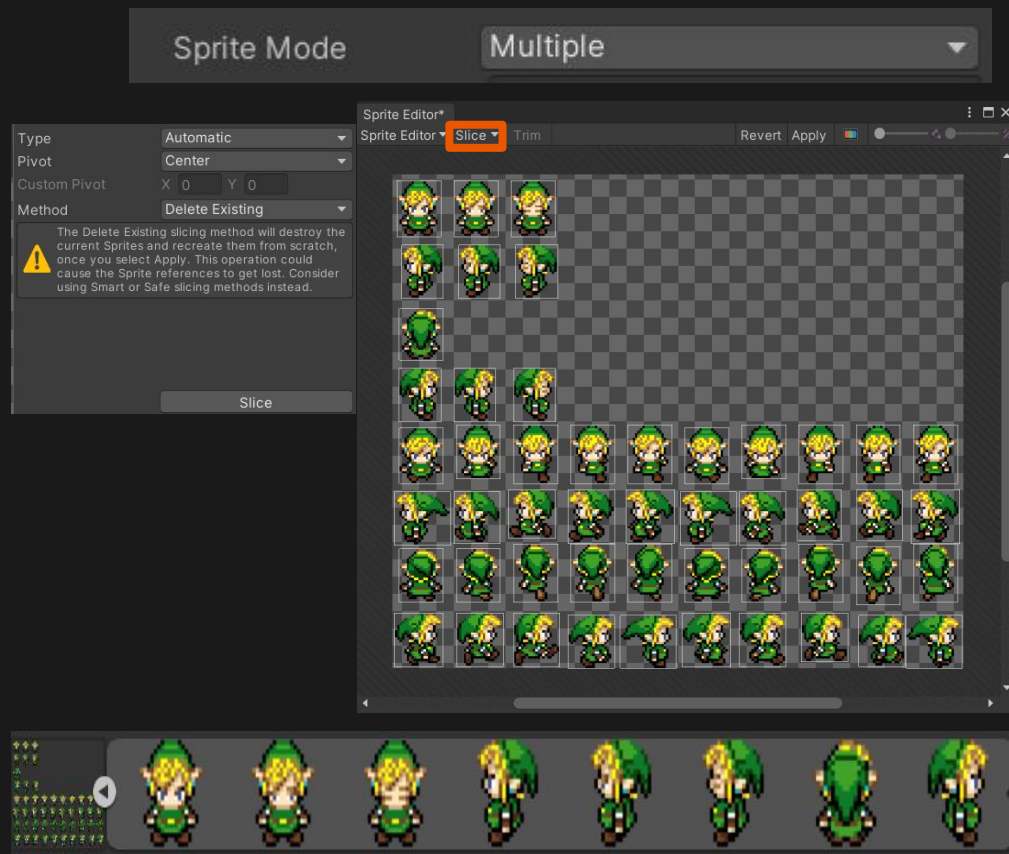


Sprite Editor

When you set your Image Import Setting of Sprite mode to Multiple and enter the Sprite Editor you will be able to slice the image into individual images.

You can do this by either Rubber Banding around the image or you can select Slice which will give you several ways to cut up the image. Automatic, by how big each cell should be or by how many cells you want in total.

Once you've separated all the images you will see that now you can see all of the individual images in the Project View.

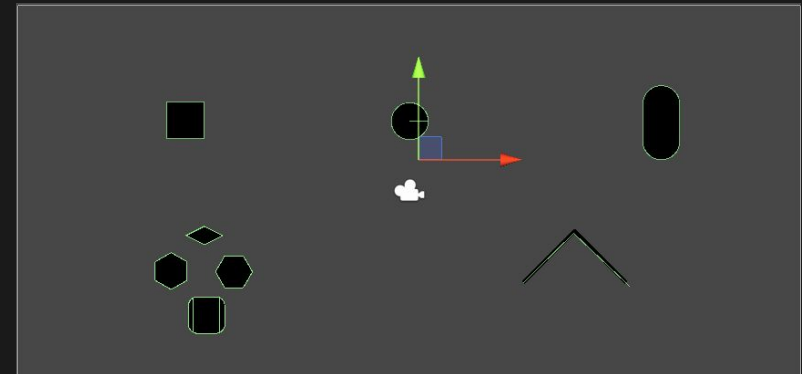
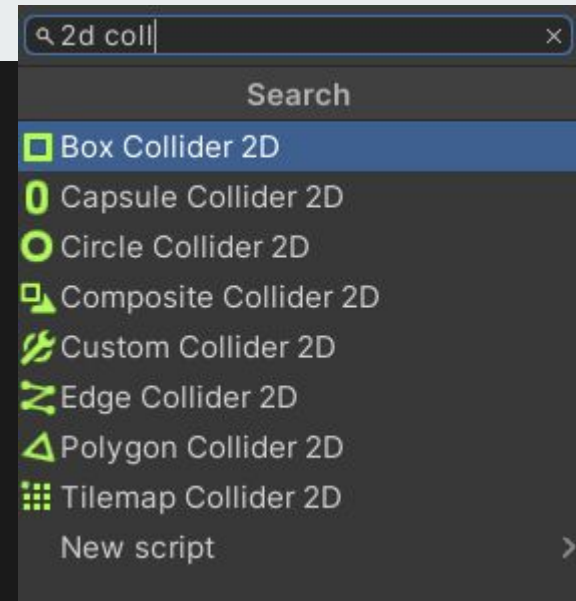


2D Collision

2D Colliders work very similarly to their 3D counterparts. With three basic one Box, Capsule and Circle with addition to Edge and Polygon collider that have exclusive properties.

Additional Resources: [Collider 2D - Official Unity](#)

Sebastian Grygorczuk - STEM Institute at CCNY



2D Edge and Polygon Colliders

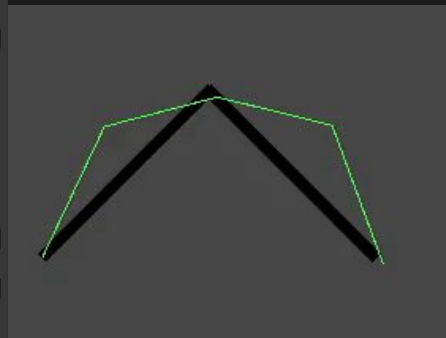
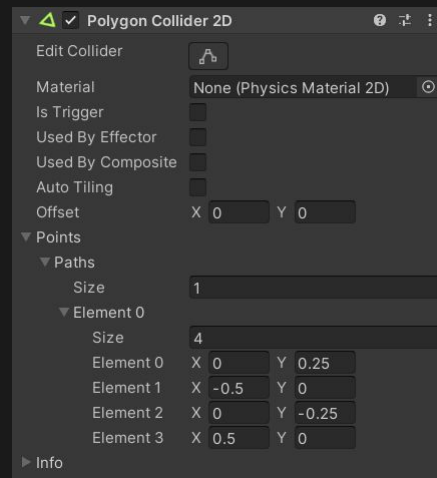
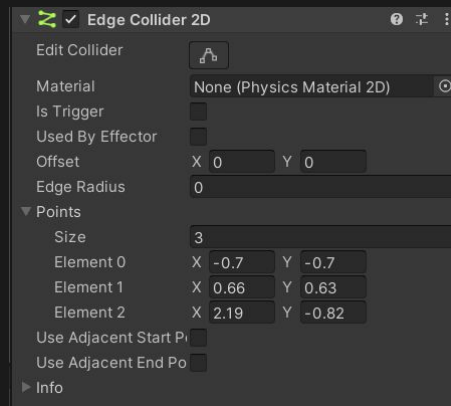


The two new interesting colliders that we have at our disposal are the **Edge** and **Polygon Colliders**.

They share the same trait that you can just draw desired shape with main difference being that **Edge Colliders** can be left open while **Polygon** have to create a loop where the last point and first point connect.

Adding a **Polygon Collider** to a given sprite will auto generate a composite that stimulates its shape.

You can edit or add the vertices when Edit Collider Mode is toggles, and if you hold **Left Ctrl** you can delete the edges.

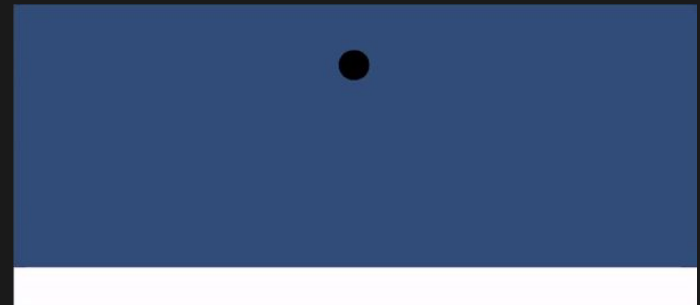
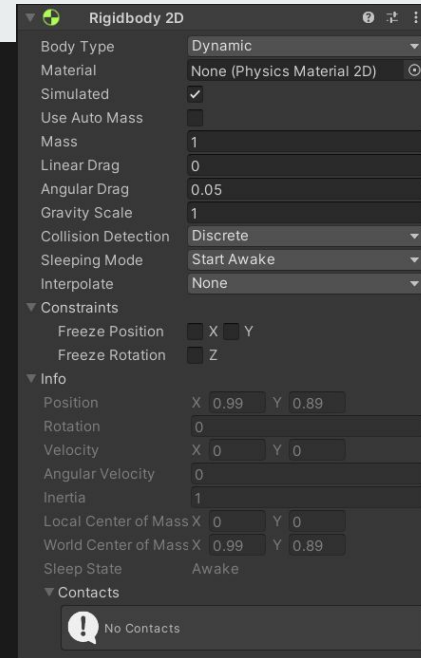


2D Rigidbody

2D Rigidbody work very similarly to the 3D one. You may have notice that unlike 3D Rigidbody the component doesn't have a toggle next to the name, this one can be controlled using the Sleeping Mode.

Additional Resources: [Rigidbody 2D - Official Unity Tutorial](#)

Sebastian Grygorczuk - STEM Institute at CCNY



2D Physics Material



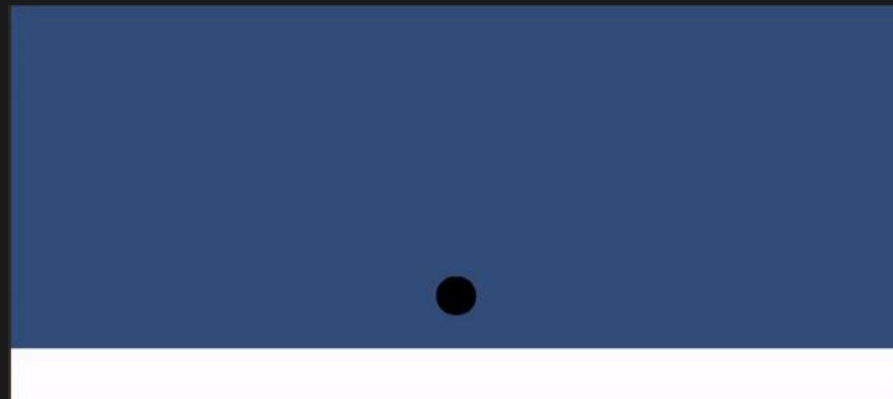
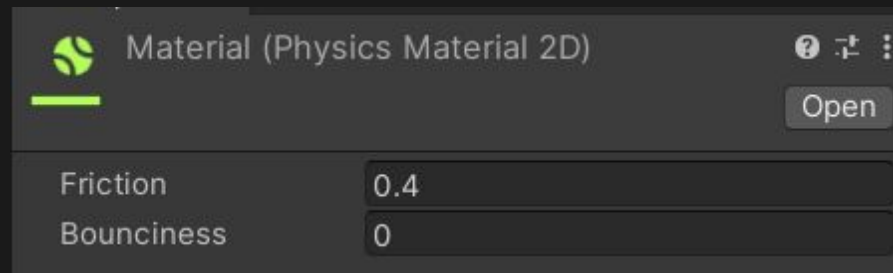
2D Object just like 3D have **Physics**

Materials you can attach to them, work

the same way with less options.

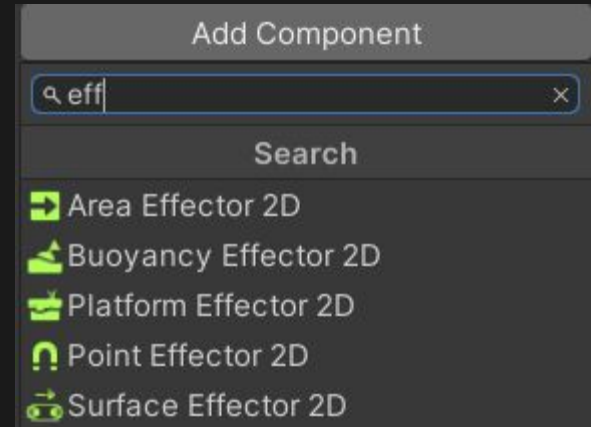
Just friction and bounciness you can

attach to the object.



2D Effectors

2D Colliders have a unique connection called Effectors, effectors are additional Physics components that continuously exert a force. There five of them, **Surface**, **Area**, **Buoyancy**, **Platform** and **Point**.



Additional Resources: [2D EFFECTORS in Unity - Tutorial](#)

Sebastian Grygorczuk - STEM Institute at CCNY

2D Effectors - Surface Effector

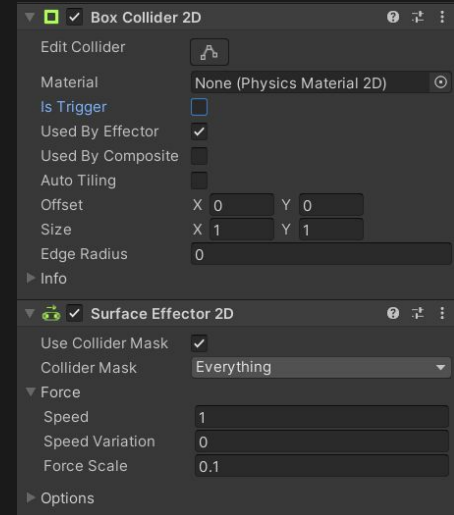


The **Surface Effector** gives a constant speed in the X axis, this is connected to the white platform and requires only the effect to be toggled on the box collider.

The **Speed Variable** will send it left or right depending on if the value is negative or positive.

The **Force Scale** is how fast the object accelerates towards the specified speed

The **Speed Variation** is any sudden upticks that occur while accelerating toward given speed.



2D Effectors - Area Effector

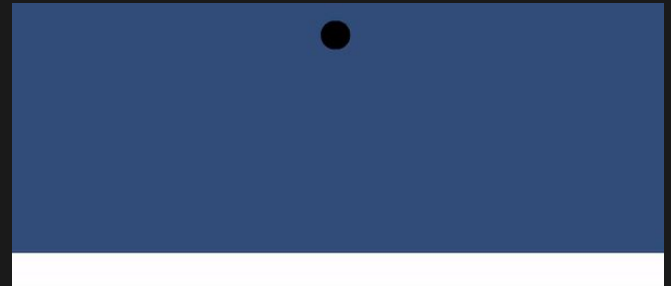
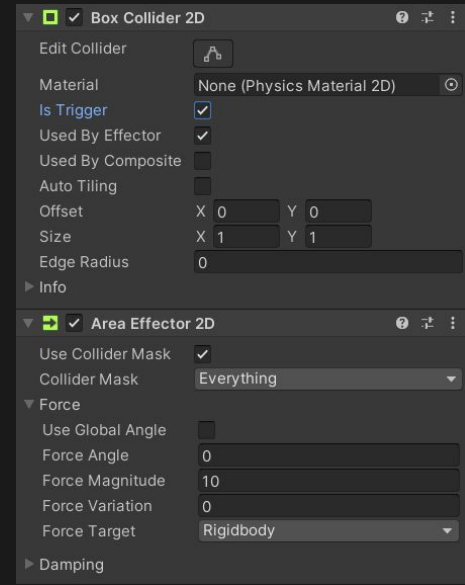


The **Area Collider** will apply a force force in a give direction to any object that enters it. This requires the Box Collider to have both the Trigger and Effector Toggled.

The **Force Angle** is which way the object will be sent, with 0 being X axis.

The **Force Mangicture** is how much force is put into the object.

And **Force Variation** is how much can be randomly added to the force, if negative how much randomly will be taken away.



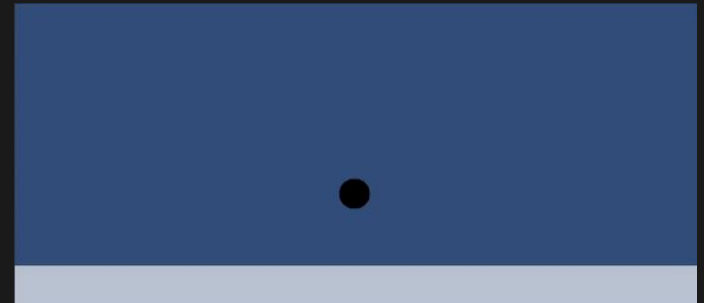
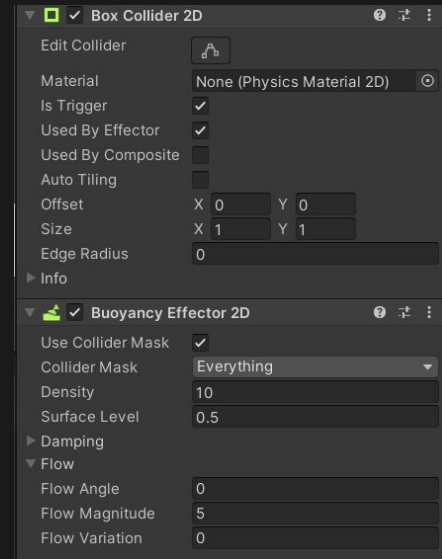
2D Effectors - Buoyancy Effector



The **Buoyancy Effector** is great at simulating a bodies of water. It requires the Box Collider to have Trigger and Effector on.

Density determines how if the object that lands in it will sink or float. While **Surface Level** will determine how much of the object will stay above water when floating.

It also has Flow menu that use the same ideas as the Surface Effector adding force to left or right of the collider to push the object.



2D Effectors - Platform Effector

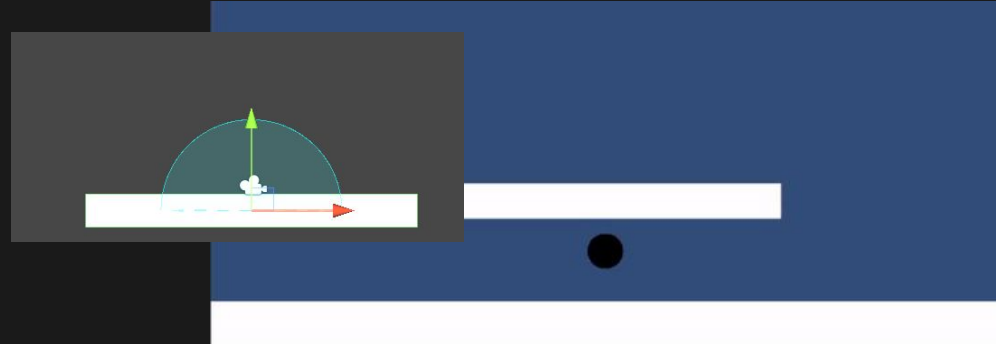
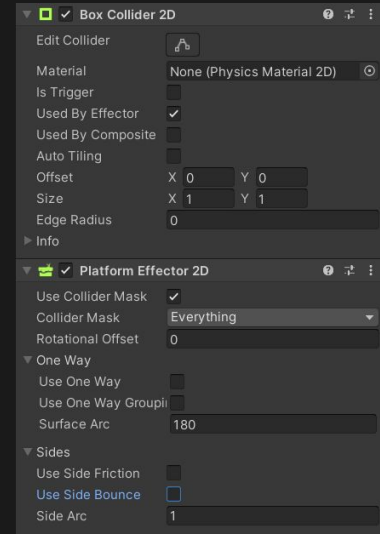


Platform Effector or One Way Platform allows us to create the effect where the play can jump through one side of the platform while keeping the other one solid.

Rotation Offset set which direction it should be facing.

Surface Arc determines which area should count as solid surface.

Sides allow you to control what the behavior should be when something collides with the platforms side.



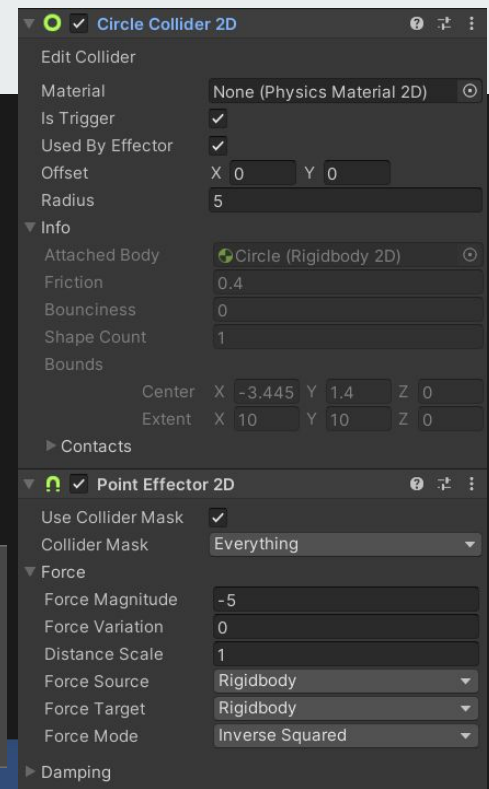
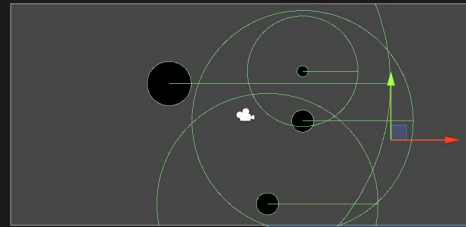
2D Effectors - Point Effector

Point Effector is very different from others, it create attraction or repulsion force between objects.

Force Magnitude tell us how strong the pull/push is.

And Force Mode allows us to determine at what rate should the force be applied.

As you see the circles have two colliders, on for actually colluding and a larger one for the effector to give range for the force to act upon.

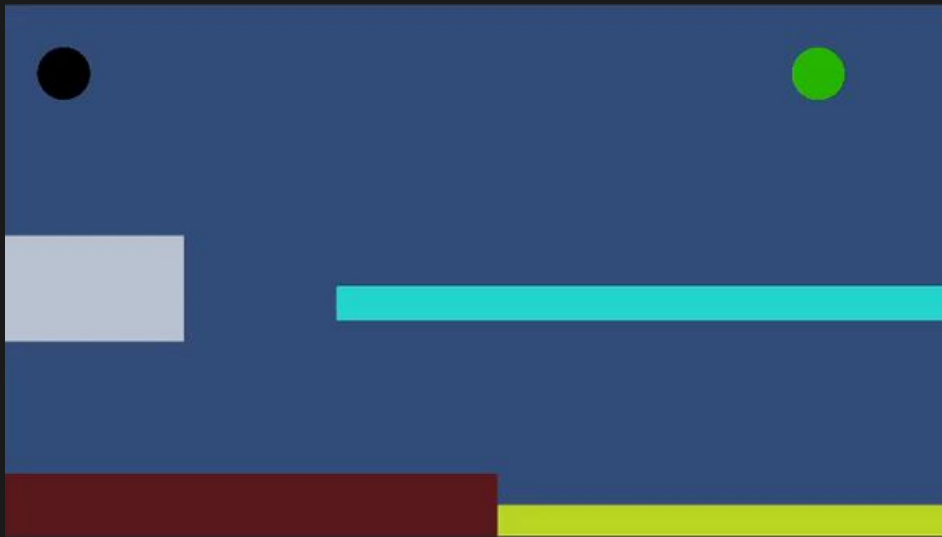


Challenge Effectors



Using the different effectors have the black circle move around the level. By:

1. Connect the **Buoyancy Effector** to the **White Rectangle** to slow down the fall of the Circle.
2. Connect **Surface Effector** to the **Red Rectangle** to move the Circle to the Right
3. Connect **Area Effector** to the **Yellow Rectangle** to send the Circle Flying Upwards
4. Connect **Platform Effector** to the **Teal Rectangle** so the Circle can pass through it.
5. Connect **Point Effector** to the **Green Circle** so the Black circle can orbit it.

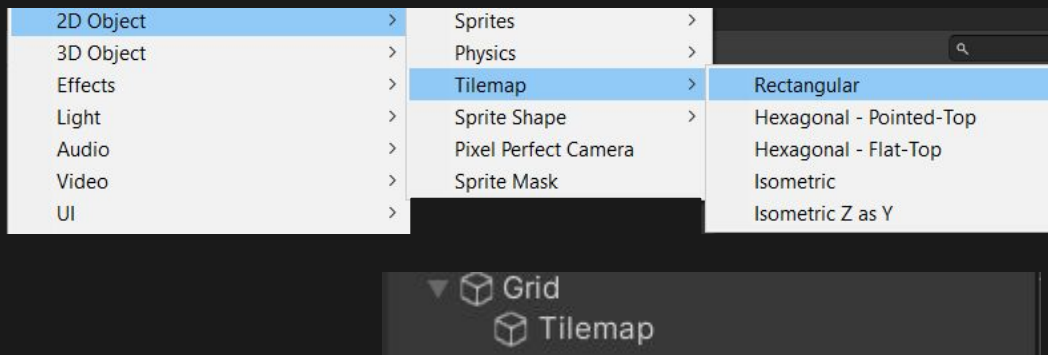


What is a Tilemap



Tilemap is a map draw from tiles. We take small sprite and line them up together to create the world of the game.

When creating a tilemap you will also instantly create a **Grid** that the tile map will be a child to. Everything that's draw on the tilemap will link to the Grid.



The Grid

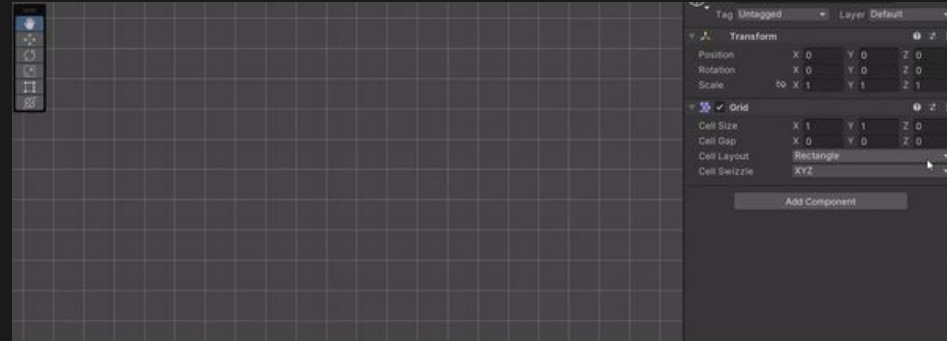


The **Grid** can be configured in three ways,

Rectuagl, Hexagonal and Isometric.

Depending on the style of game you should choose appropriate grid type. The main three being **Rectangular**, **Hexagonal**, and **Isometric**.

You also can control the size of each cell in the grid and if there should be any spacing between the tiles.



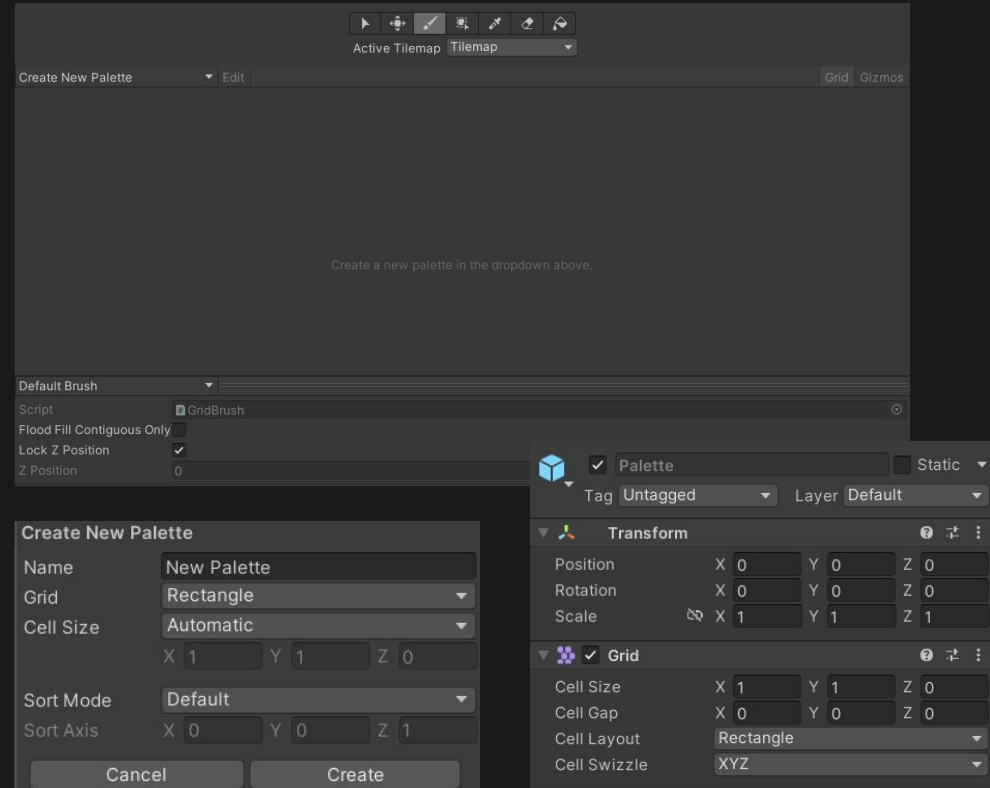
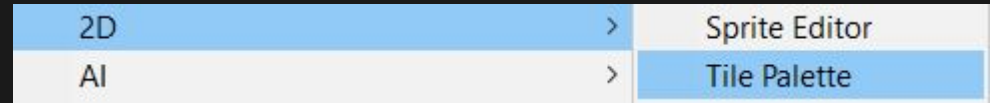
The Palette

To draw on the tilemap you will first have to open up the Tile Palette View. Go to Window -> 2D -> Tile Pallet.

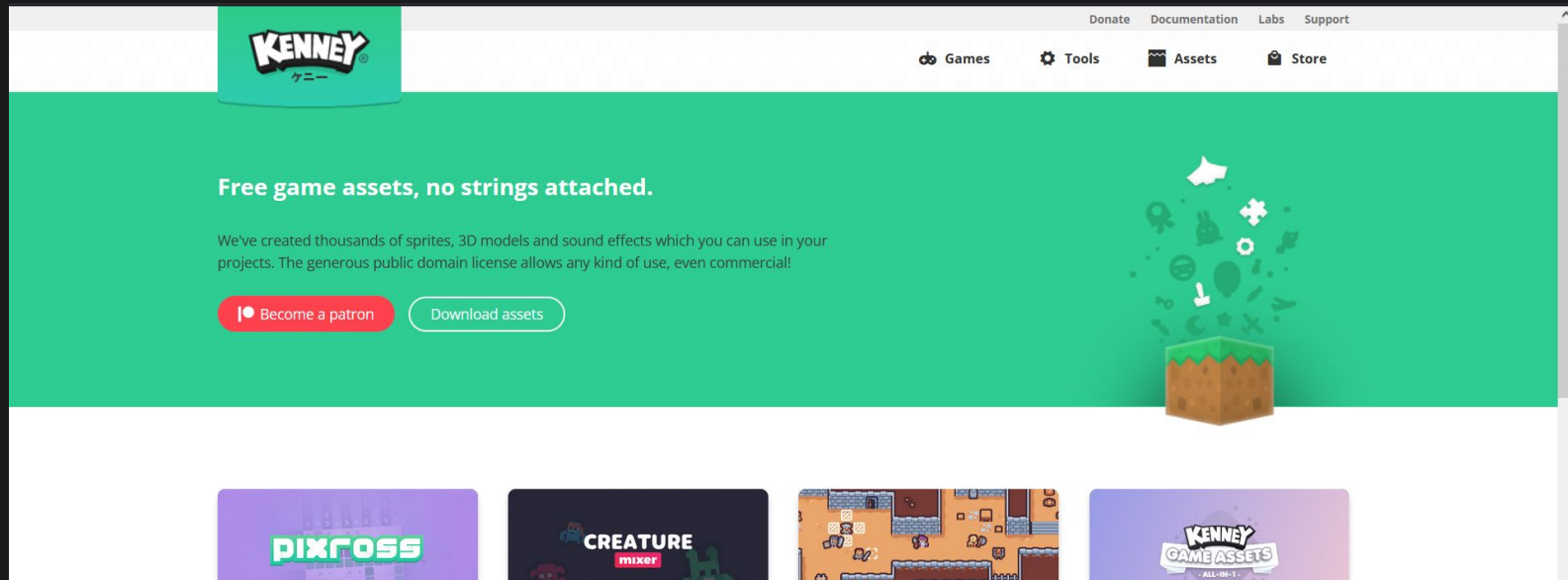
This give you the control screen that will allow you to crate a pallet.

Choose appropriate dimensions for the pallet that work with the gird you created.

This will create a Palette Prefab that will have same components as the grid.



Kenney



Kenney is a game asset creation group from which you can get assets for your 2D and 3D games.

Feel free to explore the website: <https://www.kenney.nl/>

We're going to be using [Top-down Tanks Redux Asset Set](#) for tiling

Sebastian Grygorczuk - STEM Institute at CCNY

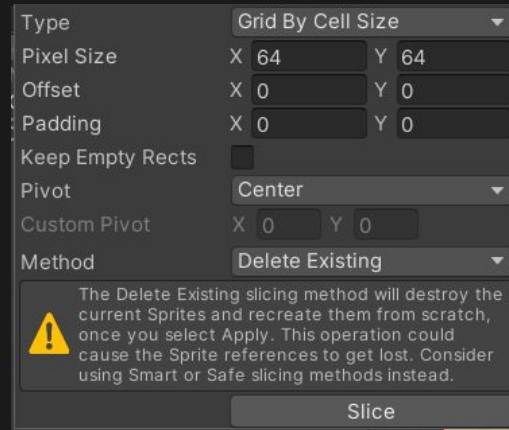
Tile Sheet



Once you have the palette set up you will need the **Tile Sheet**, the sprite that you will input into the Palette.

To make sure your image works correctly first make sure that the Pixel Per Unit is equal to the size of each cell, in this case the cells are 64x64.

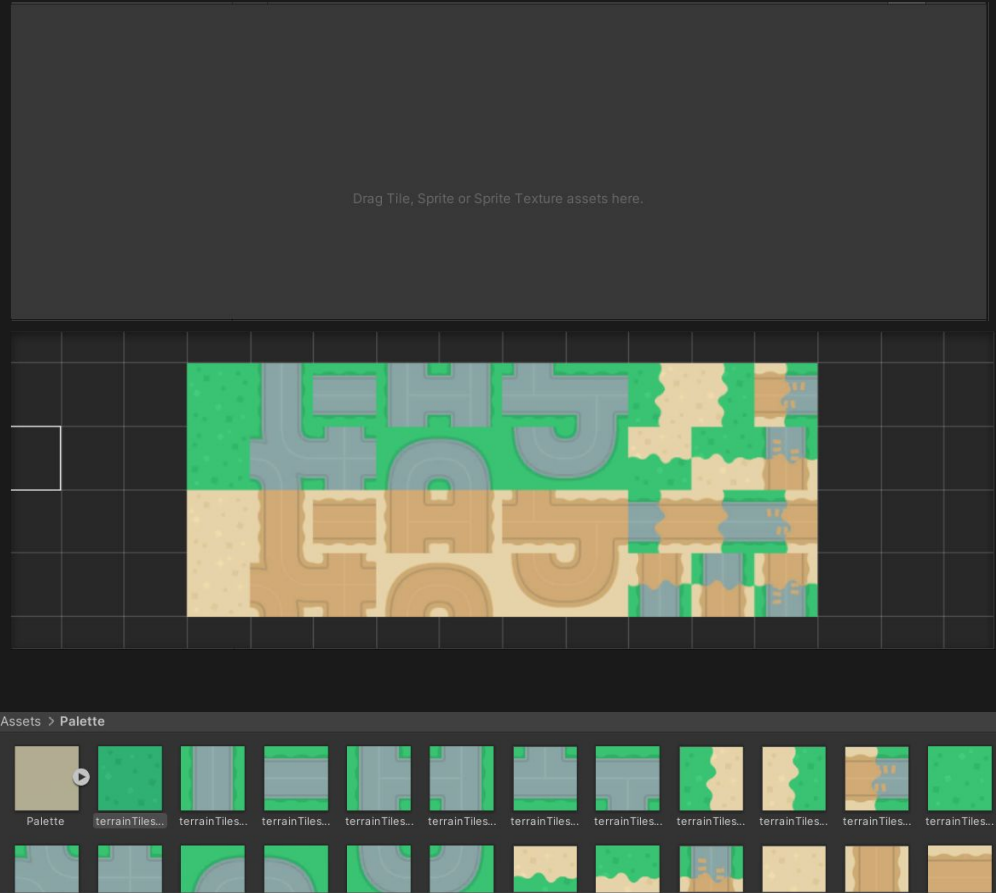
Pixels Per Unit 64



Tile Sheet



Once your Sprite is correctly configured you can drag the parent sprite to the Palette window and it will create **Tile Assets**. Have a folder ready as each cut out will have its own asset created.

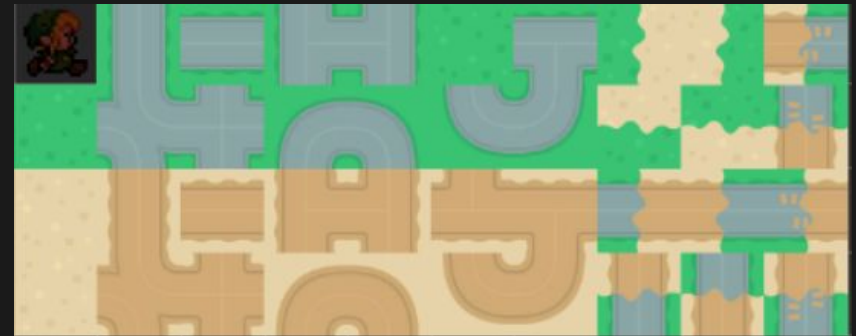
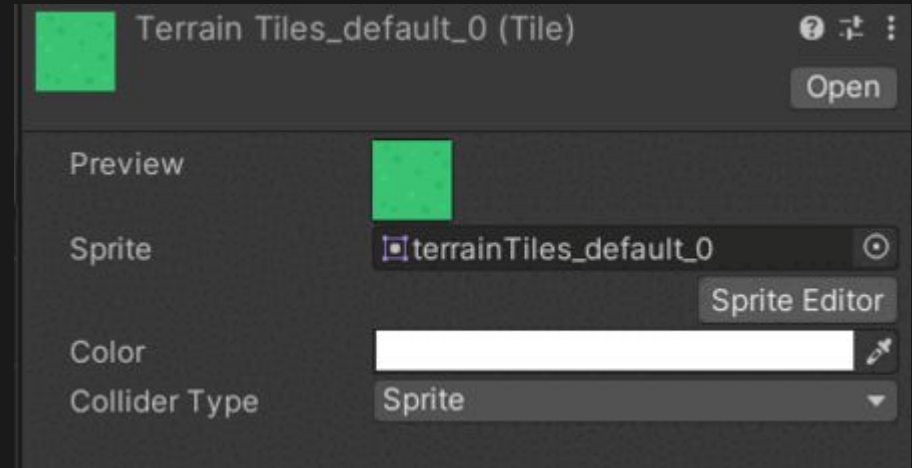


Tile Asset



The **Tile Asset** gives you control over changing what sprite is used and in what color it should be rendered.

Any changes you make will be immediately reflected in the pallet.



Challenge: Create a Tilemap

Create Tilemap from this Sprite in Sprite Folder.

- 1) Go to the **Sprite Import Settings** and change the mode to Multiple
- 2) The tiles in this Image are **16x16** so make sure that **Pixel Per Unit** is set to 16 other side the tiles won't fill up the entire square.
- 3) Open **Sprite Editor** and set it to **Slice** with 16x16 grids.
- 4) Create a new folder to store the Palette.
- 5) Create a new **Palette** using the **Tilemap View** and place it in your folder.
- 6) Drag and Drop the **Sprite Sheet** into the Palette.



Tilemap Components



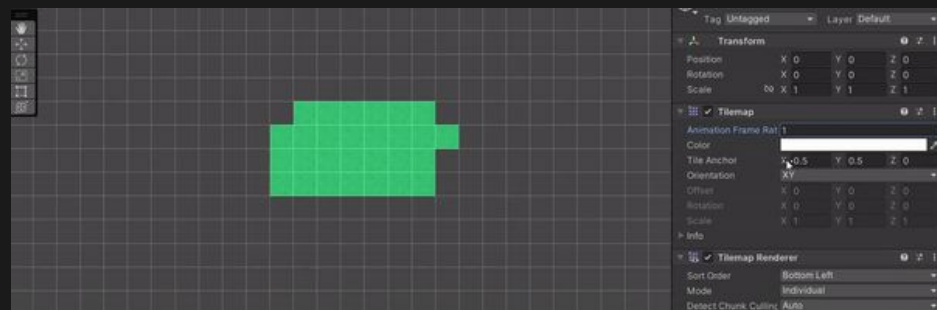
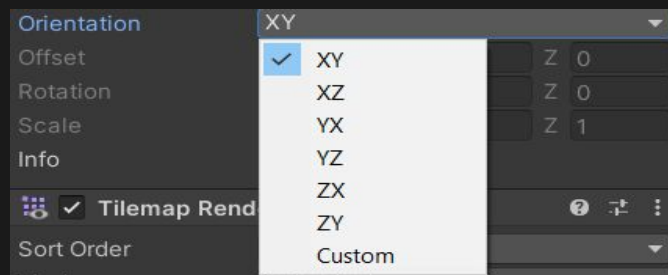
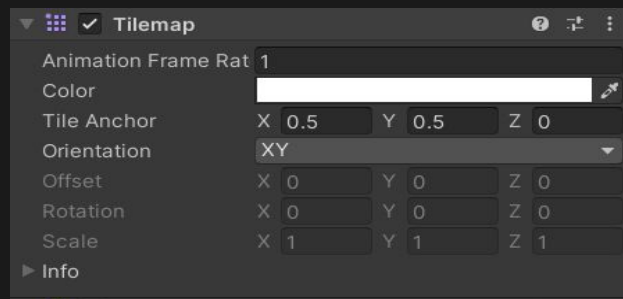
A **Tilemap** always comes with two components, Tilemap and Tilemap Render.

Tilemap acts as an internal Transform for the tiles draw on, with few rendering actions such as color and Animation Frame Rate, which we won't bother with at the moment.

You also have the power to change the plane the tiles are drawn on using Orientation.

Additional Resources: [Tilemap](#)

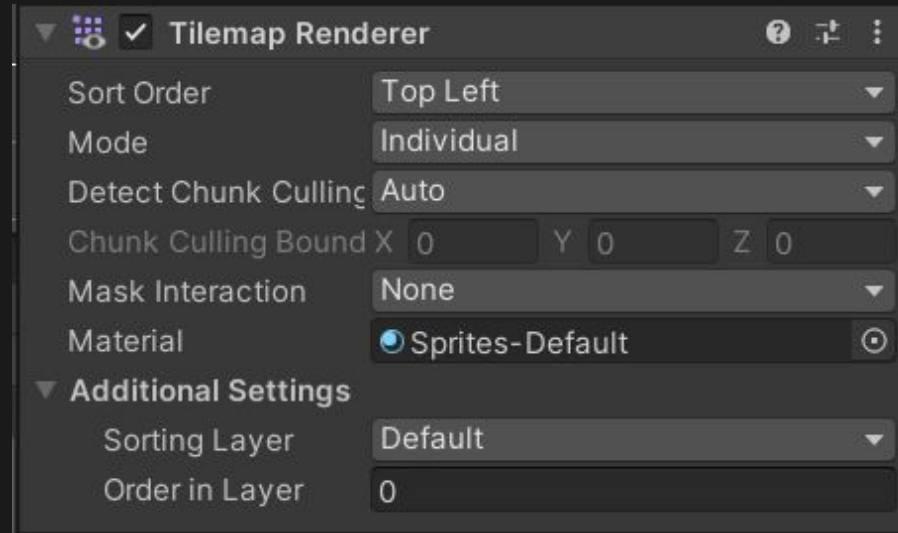
Sebastian Grygorczuk - STEM Institute at CCNY



Tilemap Renderer

Tilemap Renderer controls what get drawn and how. **Sort Order** tells us which Tiles will be drawn first, so if Top Left is Sort order that's where the drawing starts. **Mode** tells us how should they be draw, each tile on its own or in chunks. And **Detect Chunk Culling** will tell Unity how close the player has to be for it to be drawn. Think of it as **Billboarding** in 3D.

In addition to that it has our familiar Sprite controls of material, masking and layer sorting.

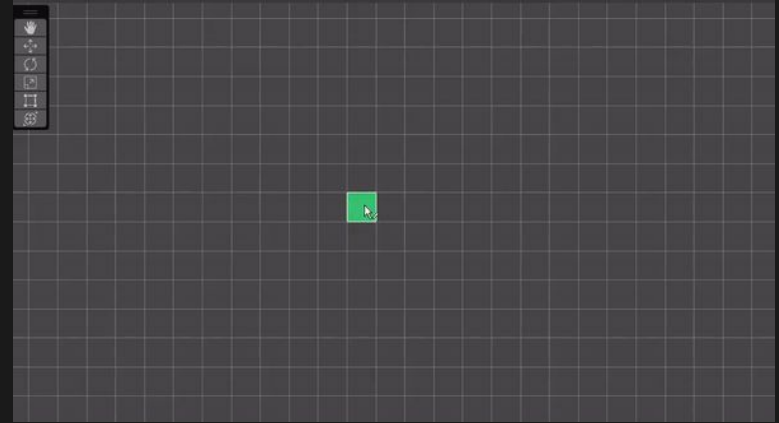


Painting



Now that we're all set up select a tile
and use the **Brush Tool** to start
painting your map.

You can also erase any tiles drawn by
either selecting the **Eraser Tool** or
holding Left Shift while painting.

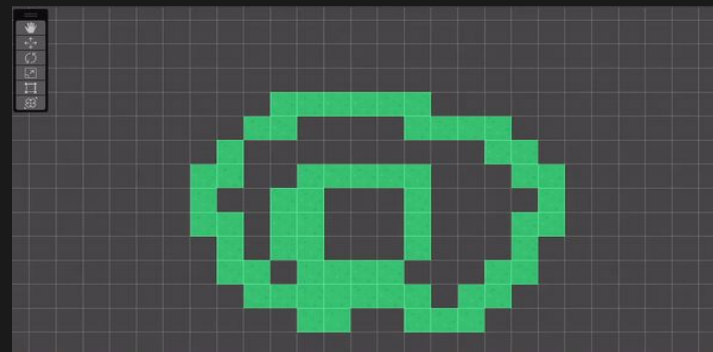
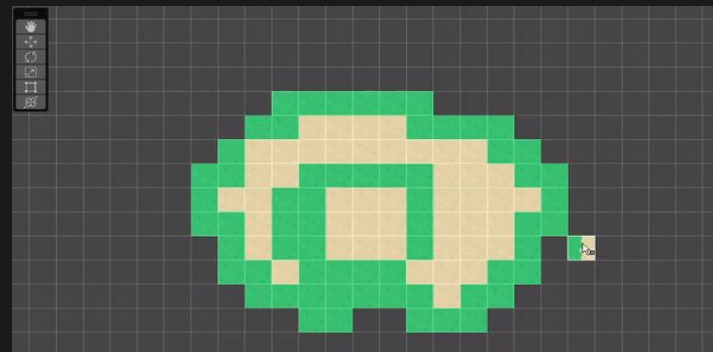


Rectangle and Fill Tool



At your disposal you also have the **Rectangle Tool** which will copy the same Tile or Tiled Area of the Rectangle Spot you create.

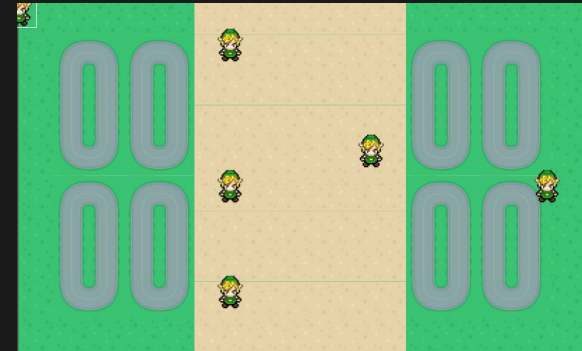
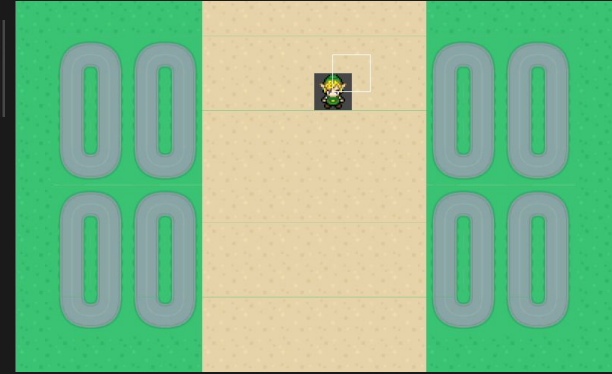
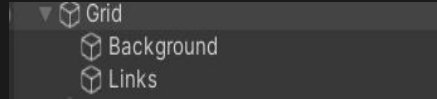
And the **Fill Bucket** which will paint all of the tiles that color as long as there's a space connecting them.



Layering Maps

If you try to place a tile on top of something you've already draw on and that tile has nothing draw on it the emptiness will be part of the tile.

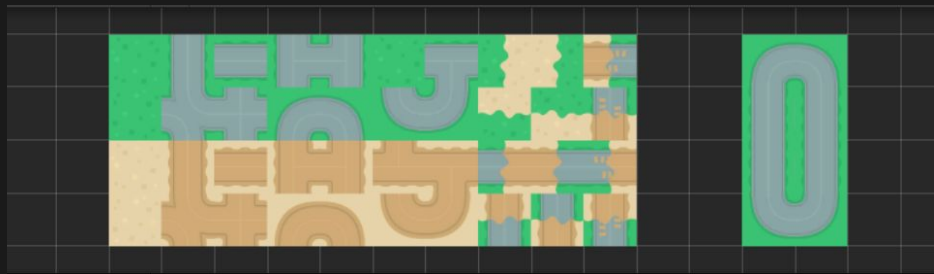
To fix this you will have to create a new tilemap who's layer will be rendered on top.



Editing Tile Sheet

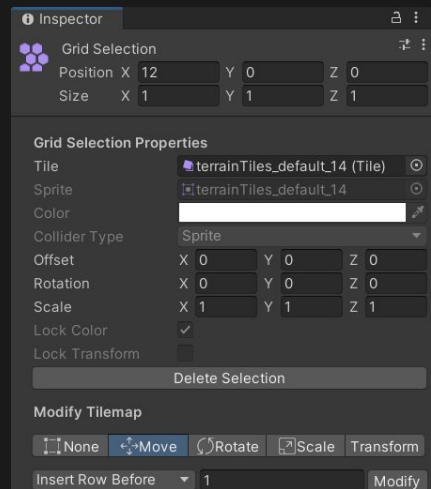


Palette Edit



Your pallet might give you a lot of option on the grid but it can take time to create complicated areas by placing the tiles one by one.

Thankfully you can create template for areas by going into the **Edit Mode** and painting in the pallet.

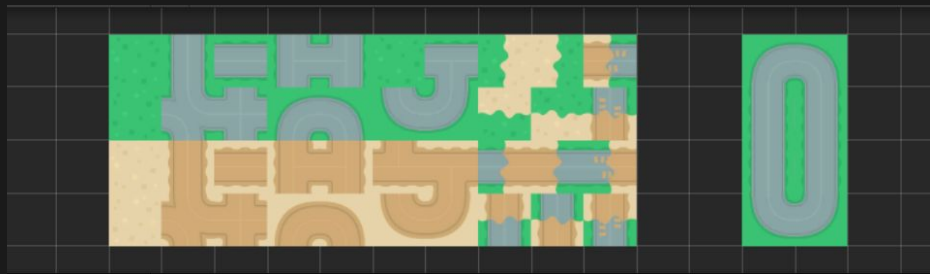


Editing Tile Sheet



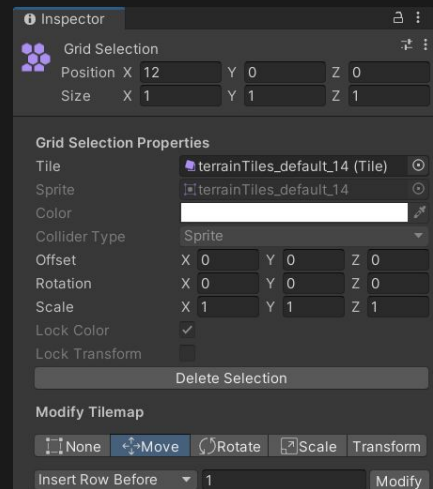
Palette

Edit



Once you're in Edit mode you can use the **Eyedropper Tool** to copy or modify the selected tile. Once you have one Selected you can use the **Brush Tool** to paint in the palette.

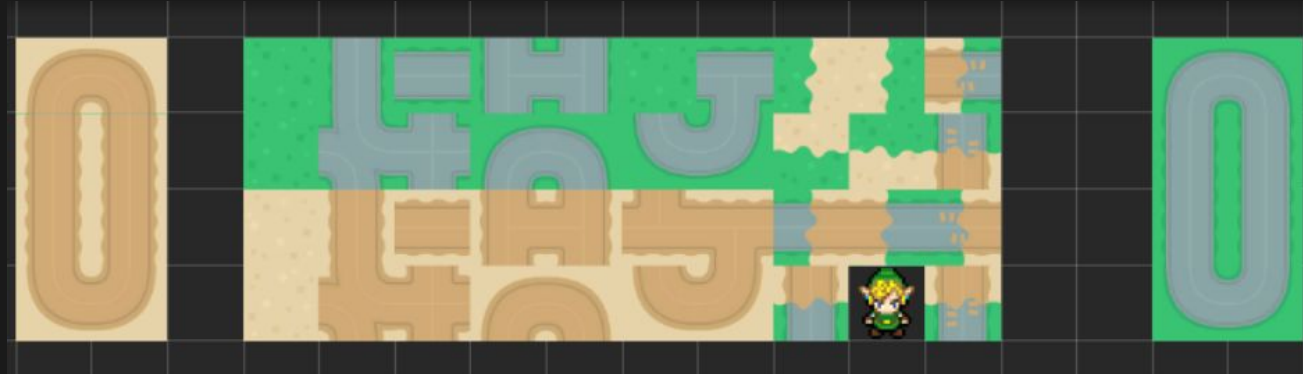
And to edit the variables of the tile you can use the **Select Tool** for the tile giving the Inspector view controls to rotate, translate and scale the tiles.



Challenge Edit Palette



Using the Edit Mode
create a round about
using the Dirt Road
Tiles.



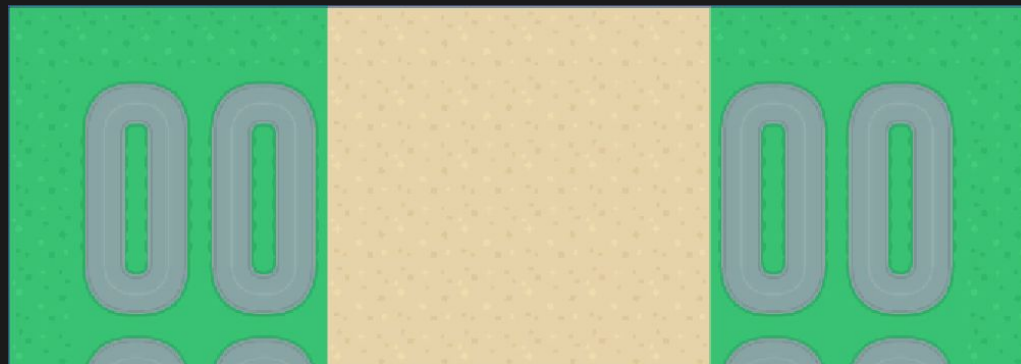
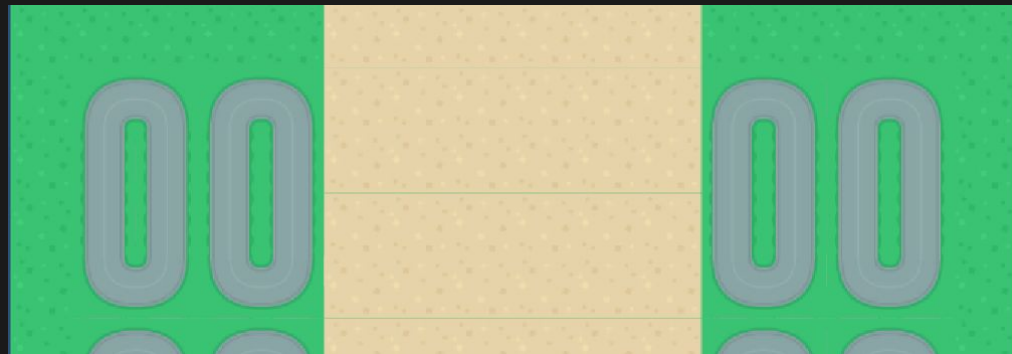
Tearing Between Tiles



You may have noticed that some of the tiles have gaps between them.

The reason behind this is that each sprite is called from the Asset Folder individual, [they may all have been connected to the big Sprite but when we split them they became individual Sprites] and the amount of work it takes for the Computer to grab the individual images and draw them on screen it creates small tears.

Sprite Atlas fixes this by connecting all of the images into one big image that will be referenced when drawing any sprite.

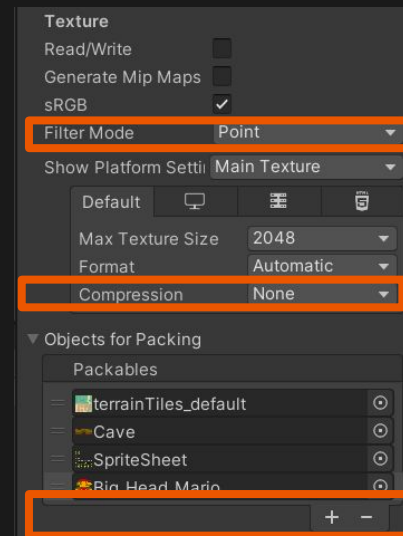
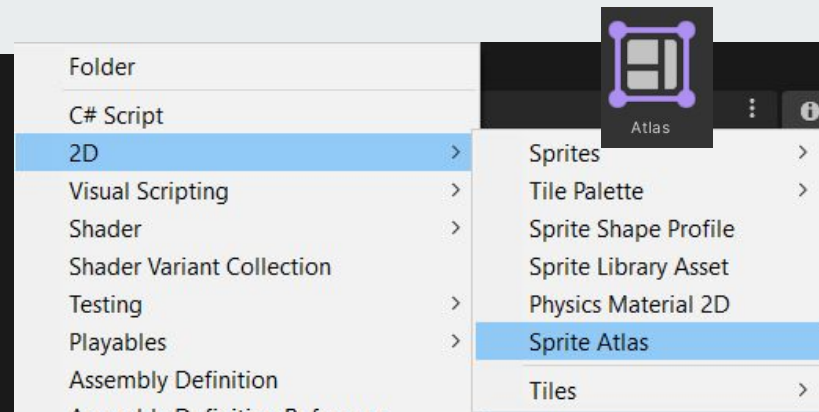


Sprite Atlas

Sprite Atlas is Game Asset, so you can create it by creating a new Asset in the Project View and going to the 2D section.

In the **Atlas Settings** you want to make sure the **Filter Mode** is set to **Point** that way the textures keep to their pixels when scaled and you want to set **Compression** to **None** so nothing becomes blurry.

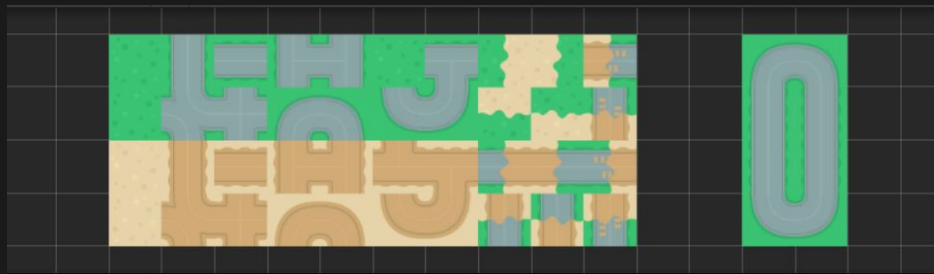
After that you can add the Sprites you want to be part of the Atlas and it will automatically generate a new Texture that Unity will be able to directly use.



Editing Tile Sheet

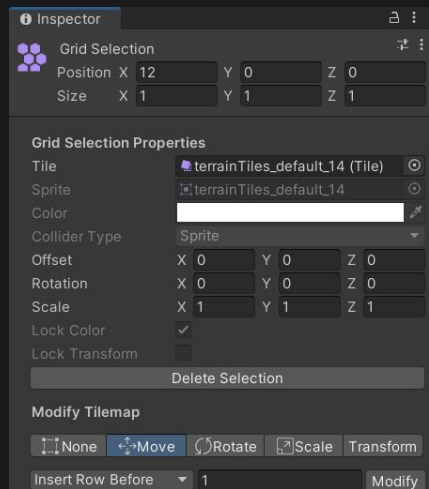


Palette Edit

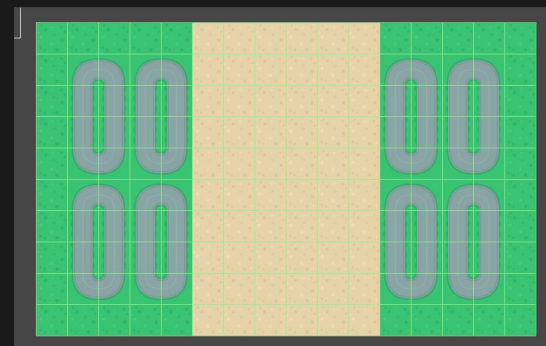


Your pallet might give you a lot of option on the grid but it can take time to create complicated areas by placing the tiles one by one.

Thankfully you can create template for areas by going into the **Edit Mode** and painting in the pallet.

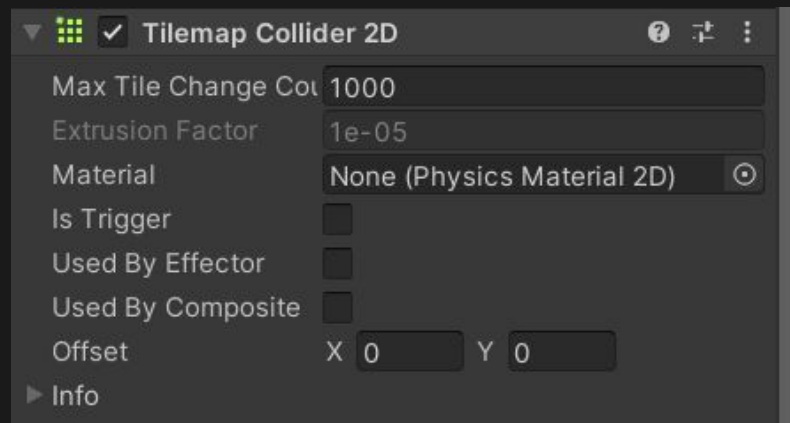


Tilemap Collision



Tilemap have a special Collider that you can attach to it allowing you to create platform or pathways for the player to interact with.

Add the **Tilemap Collider 2D** component is a start but for it to fully work it does require a **Rigidbody 2D** and since we don't want it to move in any way leaving it as **Static** is enough.



Composite Collider

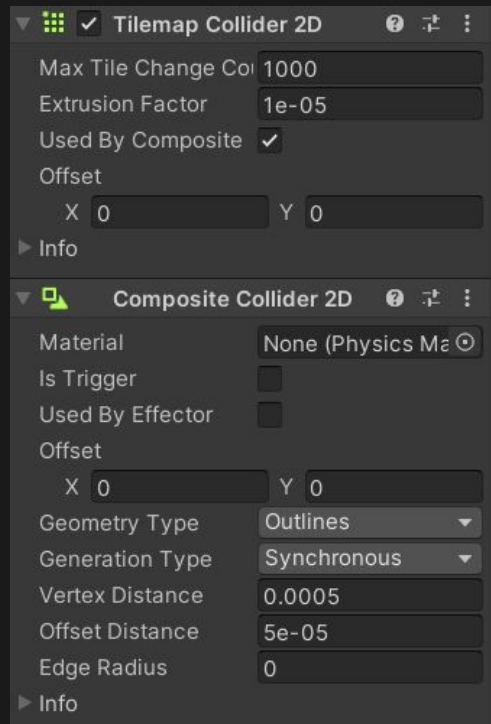


When you create a Tilemap collider you'll notice that each tile gets its own collision box.

Similar to how drawing individual tiles isn't the best simulating all of the collision boxes isn't either.

So you are making one large area it would be best to use a **Composite Collider** that will combine all of the nearby colliders into one.

This can be done to a collation of **2D Box Colliders** and **Polygon Colliders** as well.



Challenge: Build a Level



Using the Tilemap create a level for the Goomba to traverse. Use the basic color tiles to create ground and use the Roads as walls.

If you want to mess around with different tilesets download them from

<https://www.kenney.nl/>

