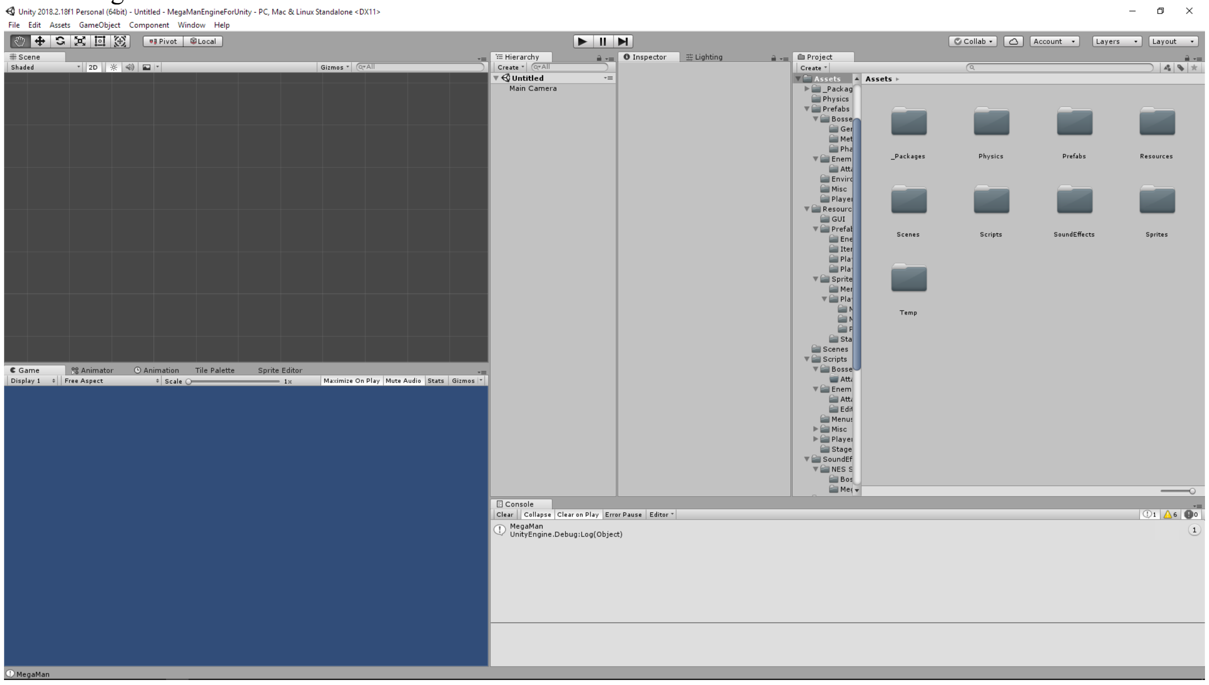
MegaMan Engine for Unity

Made by MegaChibisX

Stage Creation

Setting up a new Scene:

To create a new stage, go to **File>New Scene**, or simply press **Ctrl+N**. You should get an empty Scene now, similar to the image below.


Unity 2018.2.18f1 Personal (64bit) - Untitled - MegaManEngineForUnity - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Pivot Local
Collab Account Layers Layout
Scene Shaded 2D Gizmos
Hierarchy Create Untitled Main Camera
Inspector Lighting
Project Create Assets
_Packages Physics Prefabs Resources Scenes Scripts SoundEffects Sprites Temp
Game Animator Animation Tile Palette Sprite Editor
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Console Clear Collapse Clear on Play Error Pause Editor

```
MegaMan
UnityEngine.Debug:Log(Object)
```

Select any GameObjects already existing, and press **Delete**.

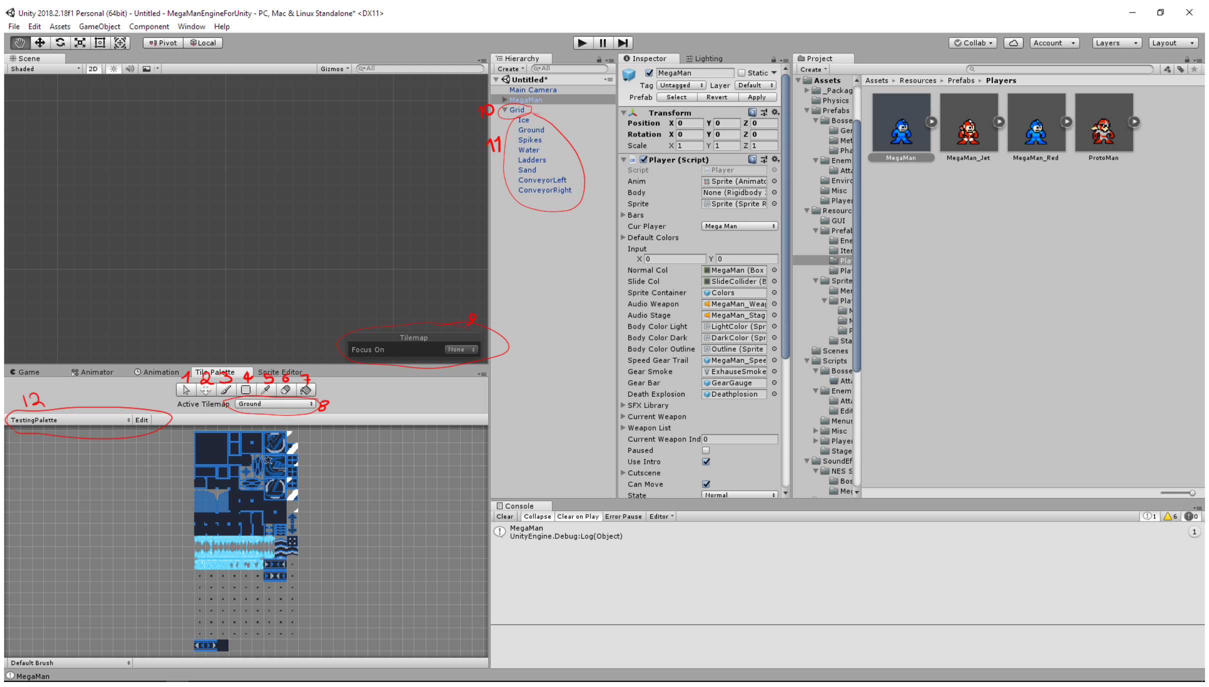
In the **Project** window, navigate to **Assets>Prefabs>Player** and drag the "**Main Camera**" into the Scene.

Go back to the **Prefab**s folder, select **Environment** and drag the **Grid** GameObject into the **Hierarchy** Window. Reset the object's position and rotation to triple zeros if needed. You can Right Click on the Title of the **Transform** component and press "**Reset**".

Finally, go back to the **Assets** folder and navitage to **Assets>Resources>Prefabs>Players** and drag a player into the Scene.
Also reset the player's position and rotation if desired.

Identifying the basic parts:

Go to the **Tile Palette** window. If there is no such window open, go to **Window>2D>Tile Palette** and drag the window next to the **Game** window.



1: **Select:** Selects an area in the Scene View.
2: **Move:** Moves a selected area**.**
3: **Brush:** Paints tiles selected from the Tile Palette into the Scene View.
4: **Rectangle Brush:** Fills an area with tiles.
5: **Tile Picker:** Selects a tile from the Scene View.
6: **Eraser:** Erases a tile from the Scene View.
7: **Bucket:** Fills a closed area in the Scene View with the selected tile.
8: **Active Tilemap:** The layer you're currently painting on. Each layer has different properties, **make sure you're painting on the proper layer!**
9: **Focus On:** Switch to **Tilemap** to focus on the layer you're currently editing. Useful to make sure you're painting on the right layer. Switch to **Grid** to focus on the entire tile map.
10: **Grid:** The parent grid, where all the tilemaps are held as GameObjects.
11: **Layers:** Each layer is held in a single GameObject inside your scene.
12: **Palettes:** The tilemaps available.

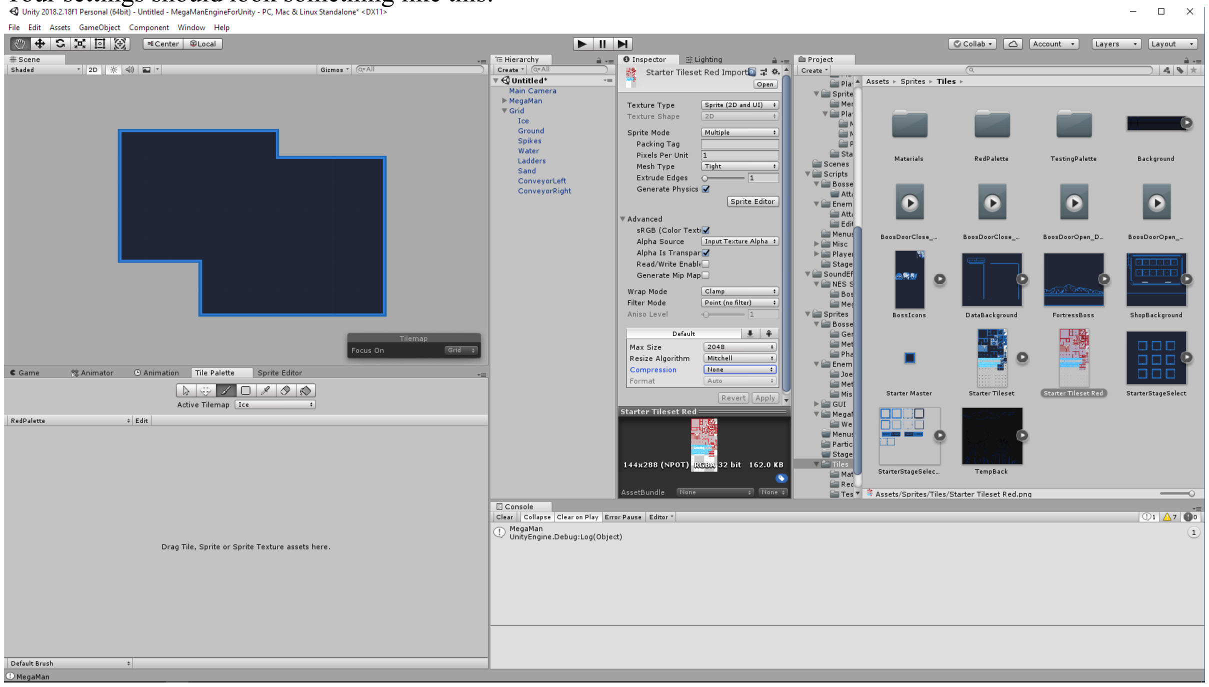
Creating your own Tileset:

To create a new palette, click on “**TestingPalette**” and select **Create New Palette**. Give it the desired name, change **Cell Size** to **Manual**, set **X** and **Y** to 16, and then press **Create**. You will be prompted to select a Folder. Go to [Unity Project Folder]>Assets>Sprites>Tiles and create a new folder. Give the folder the same name as your palette, open the folder and click the **SelectFolder** button.

Now drag your tileset image to the Project window, into Assets>Sprites>Tiles (or a subfolder in this path). You will need to change a few settings before you can use the tilemap:

- Change **Texture Type** to **Sprite(2D and UI)**.
- Change **Sprite Mode** to **Multiple**.
- Change **Pixels Per Unit** to **1**.
- Change **Filter Mode** to **Point (no filter)**.
- Change **Compression** to **None**.
- Finally, press the **Apply** button.

Your settings should look something like this:



Starter Tileset Red Import Settings

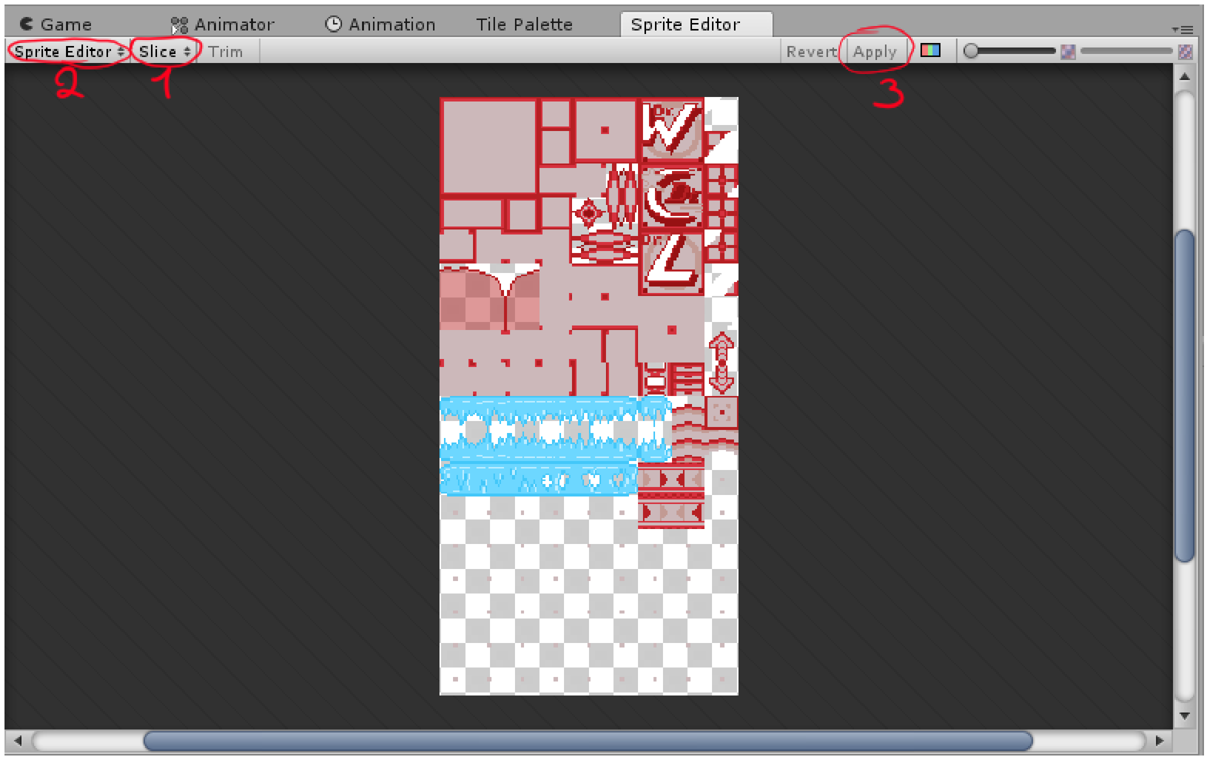
Texture Type	Sprite (2D and UI)
Texture Shape	2D
Sprite Mode	Multiple
Packing Tag	
Pixels Per Unit	1
Mesh Type	Tight
Extrude Edges	1
Generate Physics	✓
sRGB (Color Texture)	✓
Alpha Source	Input Texture Alpha
Alpha Is Transparency	✓
Read/Write Enabled	
Generate Mip Maps	
Wrap Mode	Clamp
Filter Mode	Point (no filter)
Aniso Level	1
Max Size	2048
Resize Algorithm	Mitchell
Compression	None
Format	Auto

Starter Tileset Red
144x288 (NPOT) RGBA 32 bit 162.0 KB

Assets/Sprites/Tiles/Starter Tileset Red.png

(Hint: You can make your own Tilesets over the **Starter Tileset** image already provided with the engine.)

To make and edit the tiles, click on your tileset, and press the **Sprite Editor** button.



Click on the **Slice (1)** button and change the following settings:

- Change **Type** to **Grid by Cell Size**.
- Change **Pixel Size** to (16, 16).
- If your palette has **Offset** or **Padding**, change these accordingly as well.
- Click on the **Slice** button.
- The **Apply (3)** button should be usable now. Click it to apply your changes.
- You should get an arrow pointing to the right next to your tileset in the **Project** window. If not, go back a folder and enter the **Tiles** folder again. It should now appear.

When making new tilesets, there are a few rules you should follow:

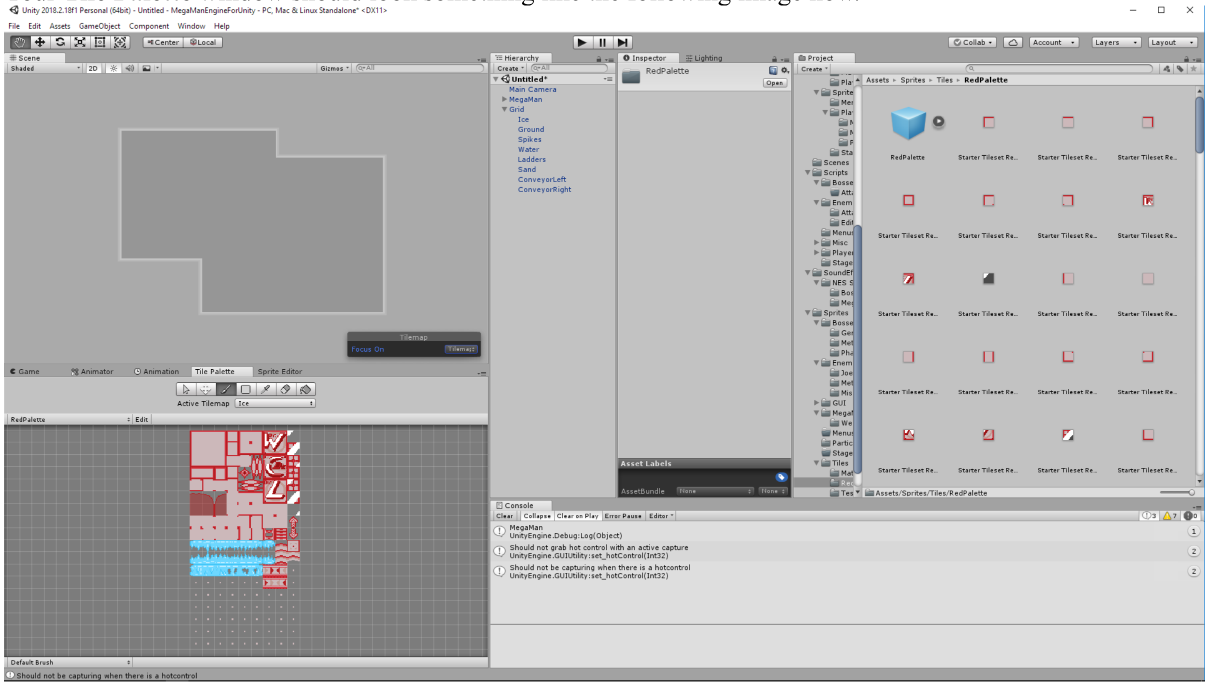
- Make sure that no panel in your tileset is completely transparent. Unity doesn't register completely transparent tiles as tiles, so if you want to add extra panels in the future, it could break your current stages and custom tiles.
- If you run out of space for tiles, only extend your tileset vertically (downwards), or risk the same problems. Make sure the new area has no fully transparent tiles.

Unity will generate hitboxes for your tiles by default. For most tiles this should work, however, some tiles will need different collisions than what Unity provides. For example, spikes should have smaller collision than their sprite shows, and their collider definitely shouldn't be a square.
To change the collider of one or more tiles, click on the **Sprite Editor (2)** button, and select **Custom Physics Shape**. Click on the tile you wish to change, click on a spot inside that tile, and drag your mouse to another spot, still inside the same tile.
You should get a rectangle over the area you drew, with 4 smaller squares (**pivots**) on the corners. You can move the pivots with your mouse, add more pivots by clicking on a line between two pivots, or remove a pivot by selecting it and pressing the **Delete** button on your keyboard.

Back to the **Tileset**, go back to the **Tile Palette** window, and select your new **Palette**.

- Select your tileset image, and drag it into the open area in the **Tile Palette** window.
- You will be prompted to select a folder again. Navigate to the folder you previously created, and press "**Select Folder**" again. (Make sure you click it while inside the folder, and not before, otherwise Unity will create a lot of files inside another folder, and you'll have to do cleanup)
- Wait for Unity to create the tiles. This might take a few minutes.

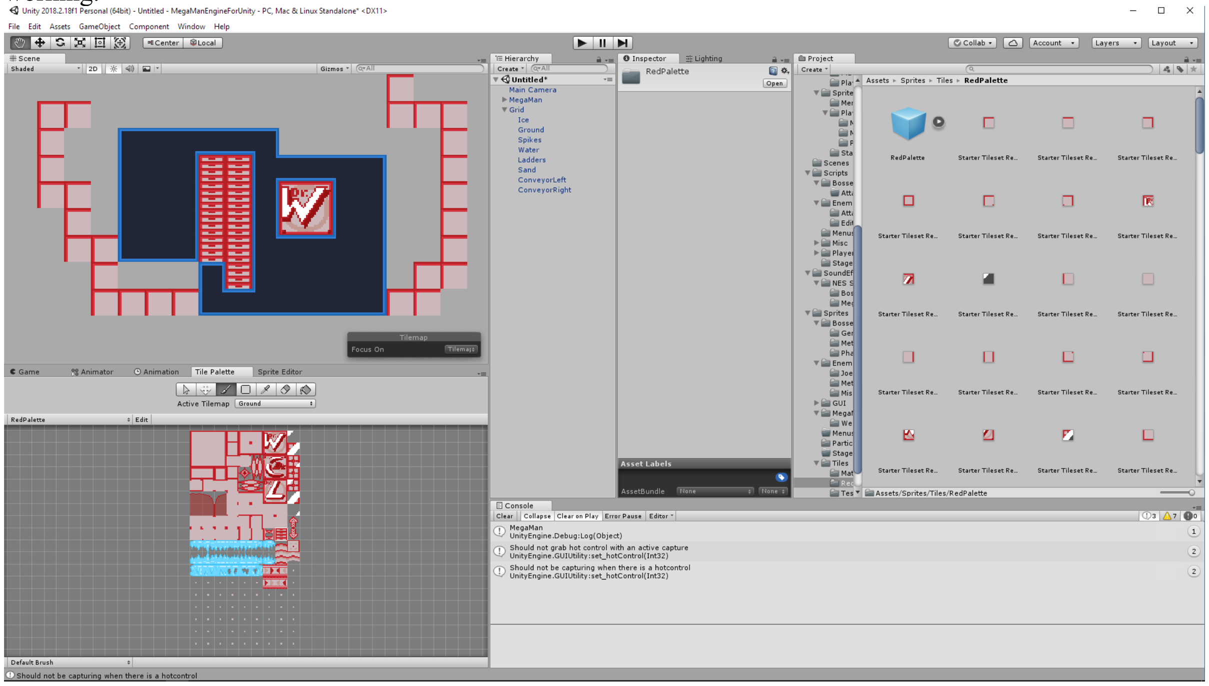
Your **Tile Palette** window should look something like the following image now.



Unity 2018.2.18f1 Personal (64bit) - Untitled - MegaManEngineForUnity - PC, Mac & Linux Standalone* <DX11>

```
MegaMan
UnityEngine.Debug:Log(Object)
Should not grab hot control with an active capture
UnityEngine.GUIUtility:set_hotControl(Int32)
Should not be capturing when there is a hotcontrol
UnityEngine.GUIUtility:set_hotControl(Int32)
```

Select the "**Ground**" layer, grab a random tile and with the brush draw on your scene. Everything should be working.



Unity 2018.2.18f1 Personal (64bit) - Untitled - MegaManEngineForUnity - PC, Mac & Linux Standalone* <DX11>

```
MegaMan
UnityEngine.Debug:Log(Object)
Should not grab hot control with an active capture
UnityEngine.GUIUtility:set_hotControl(Int32)
Should not be capturing when there is a hotcontrol
UnityEngine.GUIUtility:set_hotControl(Int32)
```

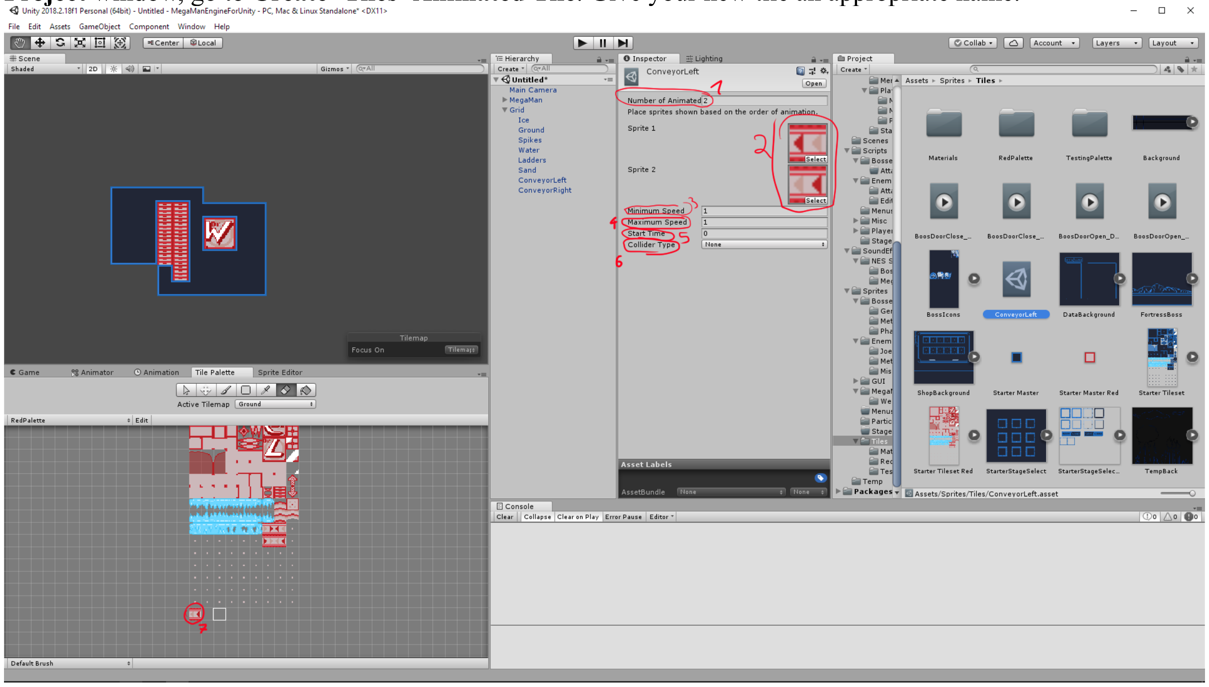
Creating Special Tiles:

Unity now comes with lots of features for 2D games on its own. The **Tile Palette** is a very useful tool, but it still lacks a few features. This project uses an Asset called **2d-extras** for any extra functionality the core Unity engine doesn't already contain, like animated tiles.

This engine was made in version 2018.2, and as such uses its appropriate package. You can find some of the newer versions by going to [Project Folder]>Version Packages.
Follow the instructions from the **Manual** about replacing the asset with a newer version, if you have updated the project to one. If your version isn't contained in **Version Packages**, you can go to the asset's page (https://github.com/Unity-Technologies/2d-extras) and download the appropriate version by clicking the "**Branch: master**" button and selecting your own version. If you have the latest version of Unity (only counting the first two numbers, like 2018.2), then **master** is the version you want to download. Otherwise, select the matching version.

Animated Tile:

To create an animated tile, navigate to the same folder your tileset image is in. Right click on an empty space in the **Project** window, go to **Create>Tiles>Animated Tile**. Give your new tile an appropriate name.



Unity 2018.2.18f1 Personal (64bit) - Untitled - MegaManEngineForUnity - PC, Mac & Linux Standalone* <DX11>

Inspector: ConveyorLeft

1 Number of Animated 2
Place sprites shown based on the order of animation.
2 Sprite 1
Sprite 2
3 Minimum Speed 1
4 Maximum Speed 1
5 Start Time 0
6 Collider Type None

Assets ▸ Sprites ▸ Tiles

Assets/Sprites/Tiles/ConveyorLeft.asset

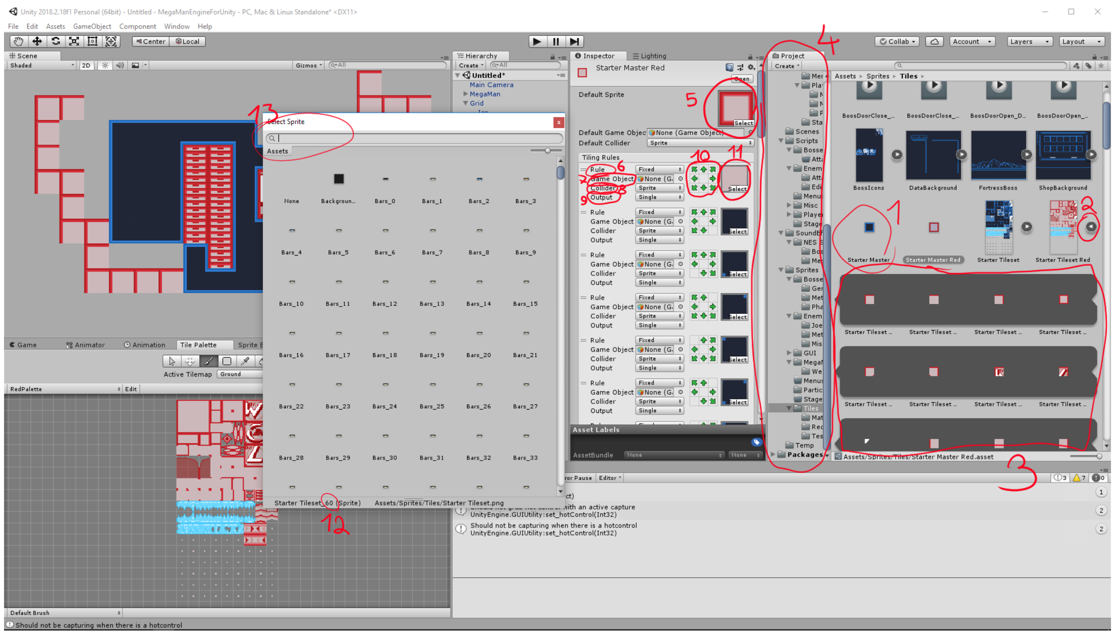
- Set the **Amount** of frames your animated tile will have (1).
- Set the different **Frames** of your tile (2)
- Set the speed of your frames, which will randomly be selected between the **Minimum Speed** (3) and **Maximum Speed** (4). Speed is measured in frames per second.
- Set the starting frame of your tile (5).
- Set the collider type of your tile (6). You can make your tile have no collision, use the collision of its tiles, or use the grid for collision.

Drag the **Animated Tiles** you've created into the folder you made earlier.

Rule Tile:

The rule tile is the tile that will likely make your life much easier if you learn how to make. Rule tiles are tiles that, depending on the tiles around them, will change their shape.

You can make a new tile by right clicking on the **Project** window and selecting Create>Tiles>Rule Tile. However, for your convenience, you can instead copy an already existing one and change the tiles with your own.



Should not grab hot control with an active capture
UnityEngine.GUIUtility:set_hotControl(Int32)

Should not be capturing when there is a hotcontrol
UnityEngine.GUIUtility:set_hotControl(Int32)

Navigate to Assets>Sprites>Tiles. Select "**Starter Master**" (1) and press **Ctrl+D** to duplicate it. Drag the new **Rule Tile** into the same folder as your tileset, if needed. You can drag it directly into the desired folder by using the small left area inside the **Project** window (4). Now rename the tile accordingly.

Open up your tileset by pressing the arrow to the right of your tileset image (2). It should now show all the tiles next to it (3).

Tiling Rules are the rules (7) the tile follows to change into the right image. Those you will normally need to set up on your own by clicking the '+' at the very bottom. They have been already set up in the **Starter Master** tile and its duplicates. For each rule there are 9 squares (10). The side and corner ones can either be empty (ignore), have a green arrow (other tile exists in that position) or a red x (no other tile in that position). The center panel is for setting special parameters about your tile, like mirroring.
You can change the collision of a rule(8), in case you don't want them to use their **Collider**.
The **Output** (9) option allows a rule to be animated, or to be randomly selected.

To change the duplicated **Starter Master** rule tile into your own, you can drag each tile into the right position from the **Project** window (4) into the icon next to the rule (11). Be sure to also set a preview tile (5).

Repeat until you have filled all the rules that you have made tiles for. There might be extra rules you haven't tiles for. For every one of these, select it, then scroll to the bottom of the **Inspector**, and press the '**-**' icon.

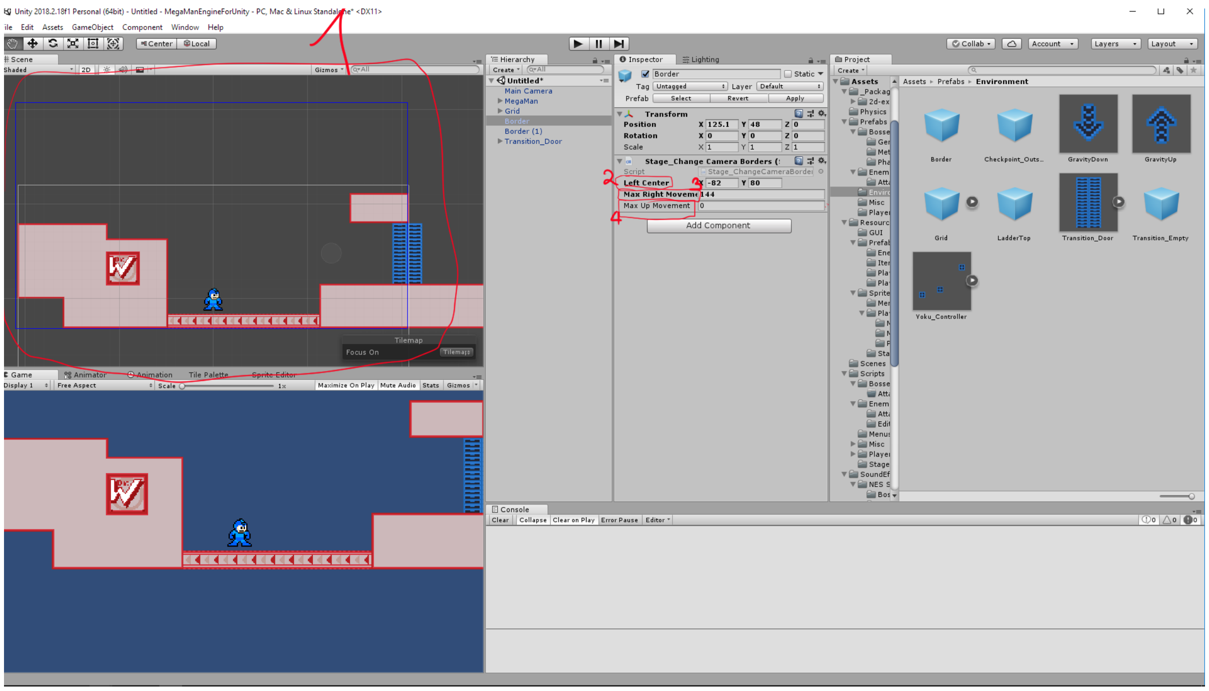
(Hint: If you've made your tileset over the one provided by the engine, you can click on the icon (11), check its index (12), type the number in the **Search Bar** (13) and get a small list of tiles. If the tileset was done right, then you can just select your tile and it should match the previous tile rule.)

Finally drag your **Rule Tile** from the **Project** window into an empty square in the **Tile Palette**. You should be able to use that tile with the brush to create stages now.

Camera Movement:

In classic MegaMan games, the camera is usually able to move between specified boundaries within a stage. In this engine, this is achieved by simply setting a **lower left corner** for the camera, as well as the amount it can move to the **right** and **up** in in-game pixels. To move between rooms, you need to use a **Border** object to define the boundaries, as well as a **Transition** to make the transition.

Border:



Inspector: Border — Tag: Untagged, Layer: Default

Transform — Position X 125.1 Y 48 Z 0; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1

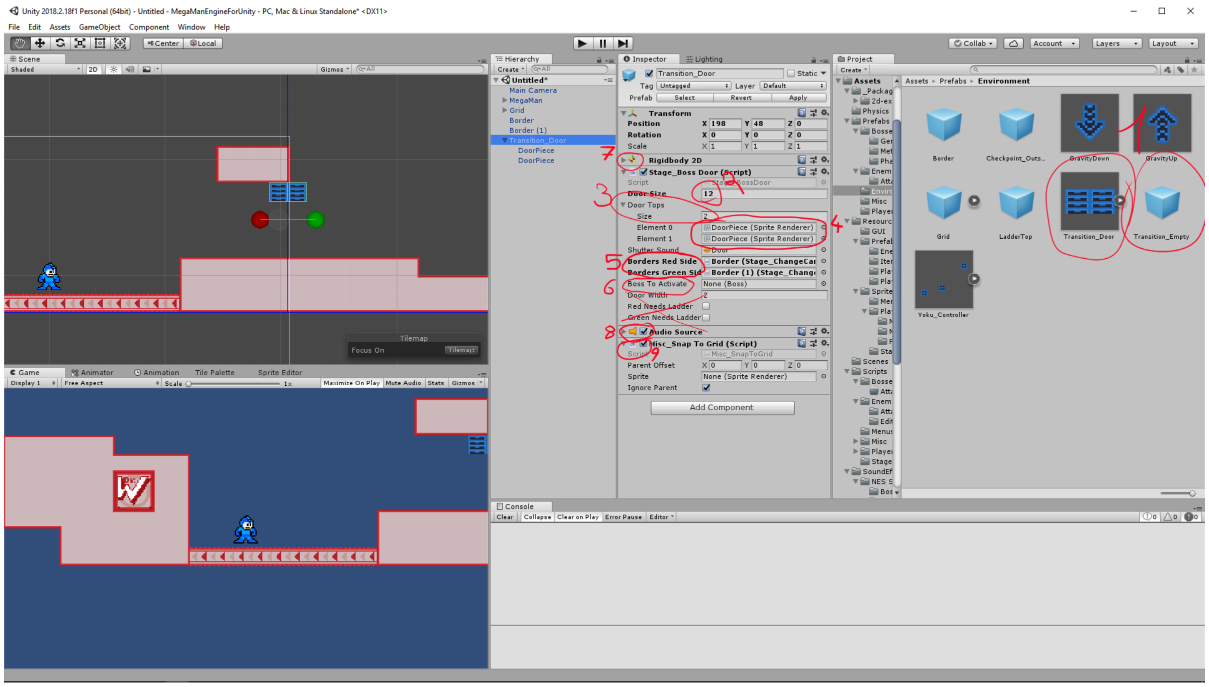
Stage_Change Camera Borders — Script: Stage_ChangeCameraBorder; Left Center X -82 Y 80; Max Right Moveme 144; Max Up Movement 0

To find a **Border** GameObject, navigate to **Assets>Prefabs>Environment** and drag a “**Border**” prefab into the scene. When selected, the **Border** will show a **blue rectangle (1)** in the **Scene** view. The rectangle shows the area that will be visible when the border is active. It will only show when the GameObject is selected in the scene.

In the **Inspector**, the **Borders** GameObject only contains a simple component, the **Stage_ChangeCameraBorders** script. From there you can set the **bottom left (2)** corner of the border, as well as as the **right movement (3)** and **upwards (4)** movement of the Camera.

At the start of the stage, assign the very first **Borders** to the **Main Camera** GameObject’s **Start Borders** field.

Transition:



Transition_Door — Tag: Untagged — Layer: Default

Transform
Position X 198 Y 48 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Rigidbody 2D

Stage_Boss Door (Script)
Script: Stage_BossDoor
Door Size: 12
Door Tops
Size: 2
Element 0: DoorPiece (Sprite Renderer)
Element 1: DoorPiece (Sprite Renderer)
Shutter Sound: Door
Borders Red Side: Border (Stage_ChangeCa
Borders Green Sid: Border (1) (Stage_Chang
Boss To Activate: None (Boss)
Door Width: 2
Red Needs Ladder: ☐
Green Needs Ladder: ☐

Audio Source

Misc_Snap To Grid (Script)
Script: Misc_SnapToGrid
Parent Offset X 0 Y 0 Z 0
Sprite: None (Sprite Renderer)
Ignore Parent: ✓

The transition doors come in different flavors, the **Door** and the **Empty** ones (1). Both work the same way.

When placed on the scene, the door will show a green and a red dot. These define the **red side** and the **green side**, for convenience. You can drag one a border from the **Hierarchy** window on each side (5), and when you touch the transition, the player and camera will be moved to the new border accordingly. The transition can also be rotated for full 360 degrees transitions.

If the transition leads to a boss fight, you can drag the boss GameObject into the **Boss To Active (6)** field. This will start the boss fight and make doors unusable. However, if you use an **Empty** transition, and there is nothing to prevent the player from going back, they will still be able to move to the previous room, even if the camera doesn't follow.

The **Door** variety of transition contains a few extra items, which are simply unassigned in the **Empty** one.

- The variable **Door Size (4)** is the height of the door, in blocks. A normal door is 4 blocks tall.
- **Door Tops (3)** holds the tiles from the very top of each door. When the game starts, the door will be extender downwards.

Furthermore the **Transitions** contain a **Rigidbody (7)** component for registering collisions, an **Audio Source (8)** for the shutter sound and a **Mist_SnapToGrid (9).**

Stage Elements:

Creating **Tilesets** has already been explained. However, the **Grid** the engine comes with has been made to have the basic functionality you will need. By going to the **Tile Palette** window and selecting the right **Layer** from the **Active Tilemap** list, as previously shown, you can change between those different functions.

The grid can be found in **Assets>Prefabs>Environment>Grid.**

Ground:
The **Ground** is the most basic element of the grid. The player can stand on tiles under the **Ground** layer.

Ice:
The **Ice** is used to mark areas where the player will slip and obey ice physics. Ice tiles need to be placed on top of **Ground** tiles for any collision to be registered.

Spikes:
The **Spikes** layer is once again easy to understand. Spikes are solid, and you can move on them while after getting hurt. Make sure to only use spike tiles in the spike layer. Only jerks use spikes that look like solid ground.

Hole:
Instant death, not solid, your bird friend can save you from it. Use any tiles you desire and make sure to put them outside of the view of the camera. The tileset provided already includes a tile with one big X you can use, but if your tileset didn't come with one, or you forgot to add one, just use spikes or something else. Preferably place them two blocks under the camera border instead of one, so the player will explode only when they are fully outside of the camera view.

Water:
The **Water** layer is the layer that tells you where water is. Go figure. Just place water tiles where you want the player to have moon gravity.

Ladders:
The **Ladders** layer allows the player to move up and down without being influenced by gravity. Come on, you've played MegaMan games before, you know the drill. The only important thing you need to note is that ladders don't normally have collision on the very top, so you need to add the semi-solid colliders manually. Go to **Assets>Prefabs>Environments** and drag "**LadderTop**" on the very top of every ladder you've placed. This might be improved in the future, but for the time being we will all have to suffer from this minor invonvenience.

Sand:
The **Sand** layer makes the player slowly sink. They can press jump to awkwardly jump out of it.

ConveyorLeft and **ConveyorRight:**
They automatically move the player left and right.

Yoku Blocks:

The yoku blocks use two scripts to ensure consistent cycles that are easy to edit. The **Stage_YokuController** and **Stage_YokuBlock** depend on each other. You can find one example under **Assets>Prefabs>Environment>Yoku_Controller**.

Yoku Controller:

The Yoku Controller controls the cycle of each set of yoku blocks. **Time Per Cycle** controls how many seconds each block will appear on screen. **Block Sound** will play the Audio Clip you select every time a set of blocks appears.

Yoku Parents is the array that holds the sets of yoku blocks and the order they will disappear with. If you only want one block to appear, you can directly assign its GameObject to one of the **Yoku Parents** fields. If you however want multiple blocks to appear/disappear at once, you will have to parent all blocks to an empty GameObject, and assign that GameObject to the field. Check the **Yoku_Controller** prefab to better understand how it all works. The **Controller** also runs on a global timer.

Yoku_Block's script uses are mainly animation and signaling the **Controller** that this is a **Yoku Block**. The **Appear Frames** and **Disappear Frames** arrays hold the appearing and disappearing animations, frame-by-frame. Assign tiles from your tileset in these. **Frames Per Second** determines the speed of your animation.

Checkpoints:

The **Stage Checkpoint** script is used to record the player's position and **Camera Borders**. If the player dies, instead of the beginning of the stage, they go back to that checkpoint. The checkpoint will be activated as soon as they touch a **Collider** marked as **Trigger** assigned to the object.

Moving Platform:

Assign a **Stage_MovingPlatform** script to a solid object to make it move around. Set the **Pivots** and **Speed** and have your platform come to life. Can be combined with other objects, like **Yoku Blocks**.

Wind Zone:

Not to be confused with Unity's wind zone, **Stage_WindZone** takes in a **Strength** and a **Direction** vector to decide the wind's position. The player will only be affected while inside the **Collider**.

Gravity Scale:

Stage_GravityScale will multiply the player's gravity by an assigned scale. They also stack. You can check **Assets>Prefabs>Environment** and drag **GravityDouble** and **GravityHalf** into the scene. When the player is inside the GameObject's **Collider**, their gravity will change accordingly.

Teleporter:

Teleporters will teleport the player to another spot (**TpPos)** in the stage. They act like transitions, so new camera borders should also be defined (**Left Center, Max Right Movement** and **Max Up Movement**).
Furthermore, a teleporter can activate a boss fight when the player goes through, or it can become disabled until a boss is dead. You can use the **Boss Rush** room as a reference.

Parallel Backgrounds:

Parallel Backgrounds can be used to give the background a sense of movement. As the camera moves, they will follow it, but they will be moving at a slower **Speed**. The **Speed** variable can be set separately for the **x** and **y** axes.
First of all, a **Follow Target** needs to be assigned. That **Target** will be the **Camera** in your scene. Drag the **Background** in the scene and drag that **Camera** into the **Follow Target** field.
For each axis, first of all the **X** (or **Y**) variable must be enabled. After that, the **Speed** variable should be set. At **1**, the background will snap to the **Target** that it follows. At **0**, it will be static. Anything between, and the background will slowly follow the **Target**. You can also experiment with other values.
The parallax effect is achieved by putting 3 images of the same background side to side and moving them with the **Stage_Background** script. To set up your own background, you have to drag the image into and parent it to the **ParallelBackground** GameObject. Duplicate it twice and move the copies to the left and right of the original image. Make sure they border each other and there are no gaps between them. Set the **Half Size** variable to be half the size of your original image.