Making 2D Games with Unity

Tutorial 2

Objectives:

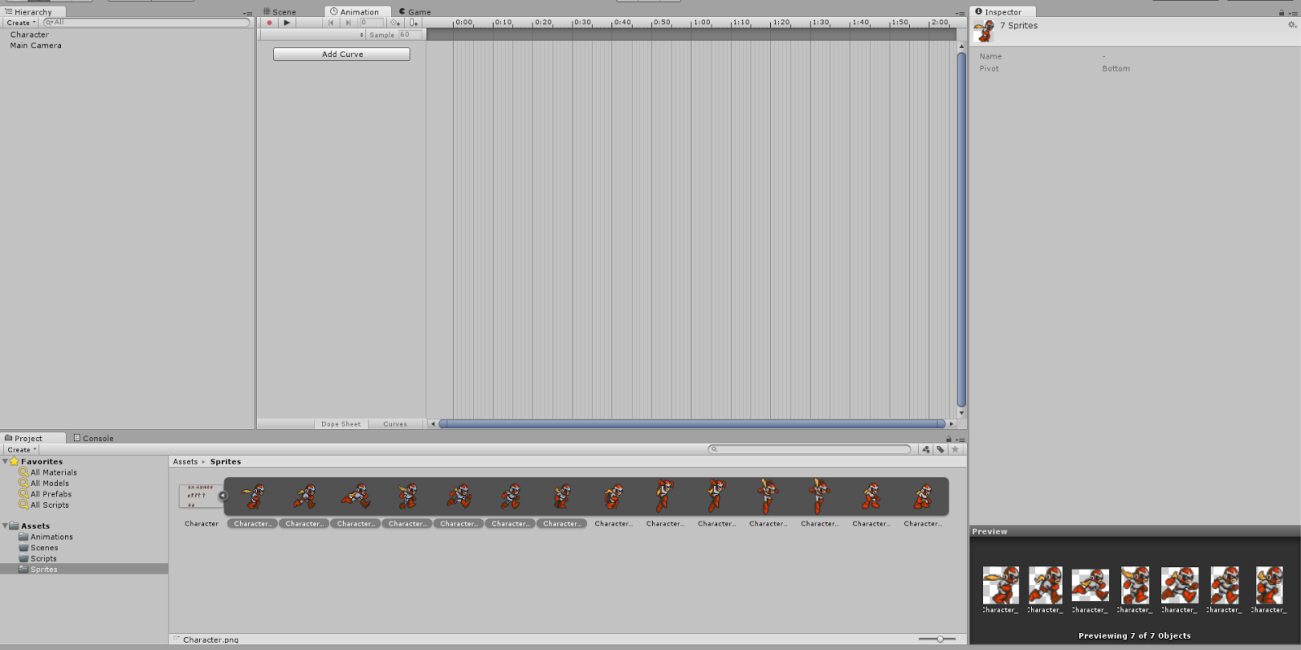
* Creating animation
* Using terrain and collision

**1. Importing/Creating Animations**

1. Create a new project called **UT-2D-2** inside a folder named **UT-2D-2.**
2. Change “Setup defaults for:” to 2D instead of 3D, and click create.
3. Once created navigate to the project tab and create 4 folders, named: Sprites, Scenes, Scripts, and Animations.
   1. Project tab>right click “Assets”>Create>Folder.
4. With the 4 folders created, import sprite maps
   1. Open the “Sprites” folder we just created via the Project tab>right click in the blank space>Import New Asset>Navigate to your sprite map>Import
   2. When the sprite map imports click on it within the project tab>sprites and you’ll see the import settings pop up on the right hand side.
   3. Switch sprite mode to “Multiple”, filter mode to “point” and format to “Truecolor”, once changed click “Apply”.
   4. Once everything has been applied, click on the “sprite editor button”.
   5. Once open click on the top left “Slice” and make sure type is set to “automatic”, minimum size fairly low (e.g. 4), pivot is usually easier when set to bottom for continuity, but for this tutorial it’s set to center, and method is set to “delete existing”.
   6. Click “slice” and you should see something like this, then click apply (top right):



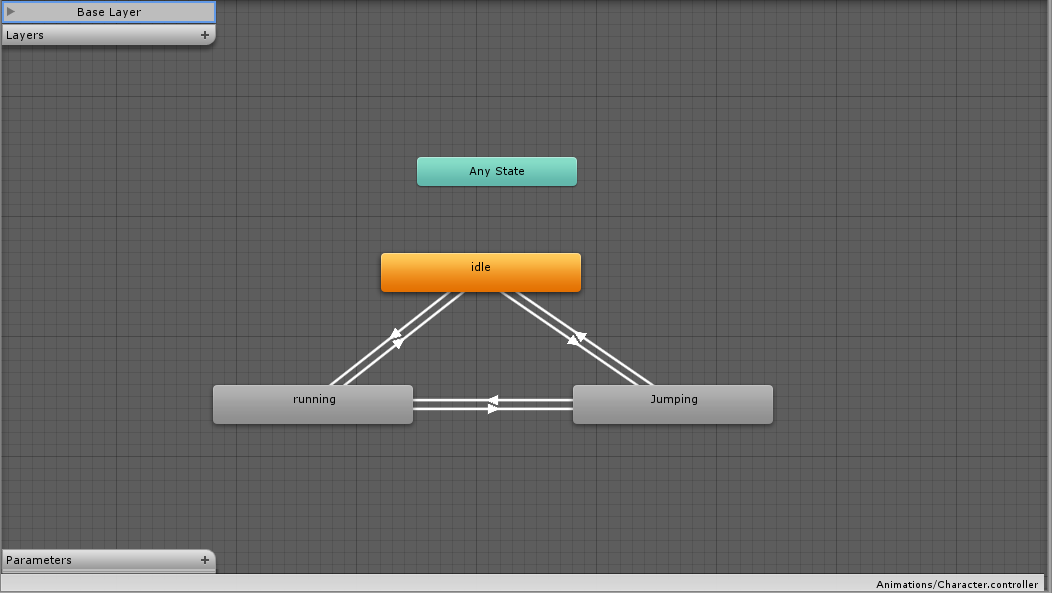
1. Now let's save the scene
   1. File>Save Scene>Open the Scenes folder we created earlier>Name it whatever you like, in this case I’ll call it “Main” for simplicity>Save
2. With the scene saved and the sprite map imported and sliced let's put the asset in the scene and create animations.
   1. GameObject>Create Other>Sprite
3. Right click the newly created sprite “New Sprite” and rename it “Character”.
4. With the Character selected you’ll notice the Inspector has 2 new components: Transform, and Sprite Renderer. In order for the sprite to animate we’ll have to add an Animator component.
   1. Add Component>Misc>Animator
5. With the animator component added, uncheck the box for “Apply root motion”.
6. Now let's group up the slices and create animations (idle, walking, jumping).
7. With “Character” selected go to Window>Animation
8. With the animation tab open and character still selected go to your sprites folder in the project tab and click the little arrow on your sprite map. This will open up all of the individual frames we sliced earlier.
9. Highlight the first segment of slices, in this case it’s the walk animation. Click on the first slice and shift click the last slice of the walking animation. Drag the selected slices over onto the animation window, anywhere along the timeline is fine.



1. A popup will appear, this is our first animation the walk/run cycle so let's call it running and save it into our Animations folder.
   1. Click “Save”
2. The 7 frames/slices will appear really tight together on the timeline, to fix this lets change the sample size to 7 (the amount of slices/frames we have). \*Located to the left of the slices in the timeline
3. Now let's repeat that with the other two states (idle and jumping), however since the running timeline is already open we’ll have to create it a different way.
   1. Next to where we changed the sample size click on “running”>Create new clip
   2. Similarly to how we did it before, change the name to jumping and save it in the animations folder.
   3. Once saved the timeline will be empty, so click on the jumping frames/slices and drag them onto the empty timeline. Again changing the sample size to the amount of frames/slices you dragged over. In this case there was 5 frames/slices for jumping.
4. Do step 16 again however this time for the idle state (2 frames).

**2. Implementing animations**

1. With the 3 states created lets implement them.
2. Window>Animator, again make sure you have “Character” highlighted under the hierarchy tab.
3. Now let's set up the animations so we can see them in action
   1. You’ll notice 4 rectangles: Any State (blue), running (orange), jumping, and idle.
   2. Running is orange because it was the first one we created, meaning it’s the default animation. However in our case we want the idle animation to be the default since when we aren’t moving we don't want our character to be running on the spot.
   3. Right click the “idle” rectangle>set as default
   4. With idle set as the default animation let's make animation transitions so later when we press a key the run/jump animation starts and when we stop the idle animation takes over again.
   5. Click and drag the running, jumping, and idle rectangle away from each other to form a triangle (screenshot below). This is solely for organization.
   6. Right click on idle>make transition>click on running, and Right click on idle>make transition>click on jumping
   7. Idle should have 2 arrows, one pointing at running and the other a jumping. Do the exact same process but from running and jumping perspective.
   8. It should look like this:

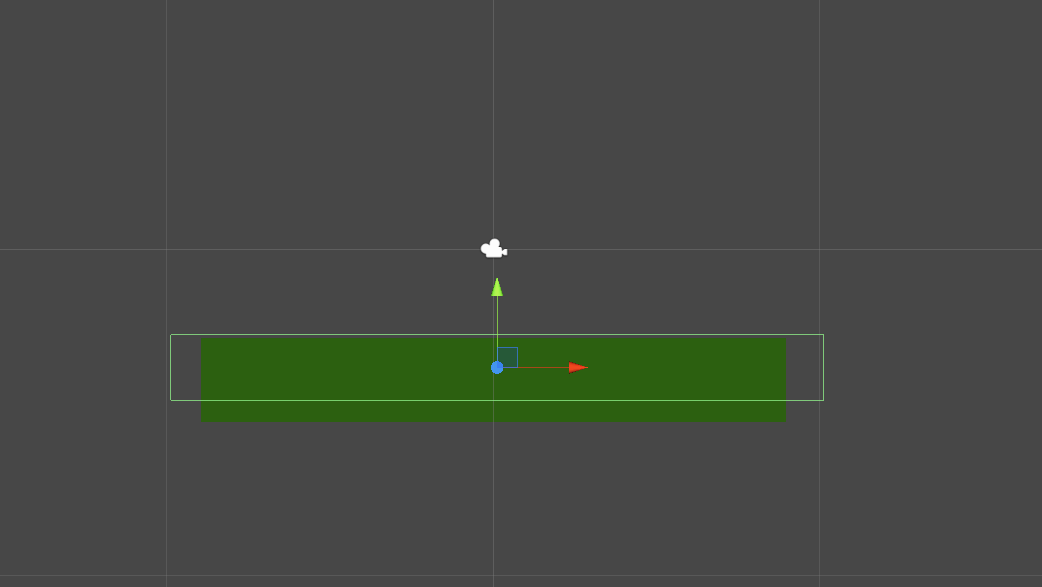


1. With the animations linked lets set some parameters so we can transition between the states (animations).
   1. Click the plus sign next to the parameters tab in the Animator window>Float>name it moveSpeed
   2. Click the plus sign next to the parameters tab in the Animator window>Trigger>jump
2. With the two parameters implemented lets apply them to the transitions, to do so click on an arrow and on the right hand side under the inspector tab there’s a section called “Conditions”. To add a condition click the plus sign, to remove a condition click the minus sign.
3. With that being said look at the table below and add/remove conditions so that all the arrows are identical to the table.

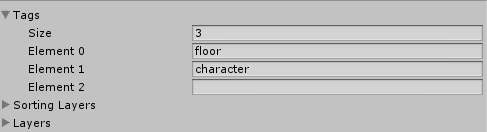
|  |  |  |
| --- | --- | --- |
| From | To | Conditions |
| Idle | Running | moveSpeed > 0.1 |
| Idle | Jumping | jump |
| Jumping | Idle | Exit Time 1.00 |
| Jumping | Running | Exit Time 1.00 |
| Running | idle | moveSpeed < 0.1 |
| Running | Jumping | jump |

**3. Terrain and Colliders**

1. Import the ground sprite
   1. Project tab>Sprites>Right Click in space>Import new asset>locate the sprite you created
2. Drag the sprite into the hierarchy tab
3. Position the ground so it fits properly in the scene (viewing from scene and game viewports)
   1. Try changing the x,y,z to 0,-4,0 respectively and increasing the x scale to 1.75 to make it a little longer.
4. Once properly placed in scene add a Box Collider 2D to the gameObject ground
   1. Click on the ground gameObject in the hierarchy, under the inspector click “Add Component”>Type in Box Collider 2D
5. With the floor/ground created we need to create a separate collider to tell us when the character touches the ground. (So we know when the character can jump again).
   1. GameObject>Create Empty>Name it floorCollider or groundCollider.
   2. Under the inspector>Add Component>Box Collider 2D
   3. Check the box for “is Trigger” and manipulate the size until it is practically identical to the outline of the ground. I used size values X:20 and Y:2. With the size figured out drag the box collider in scene view on top of the floor and make sure it goes slightly higher than the floor. This is to allow the floorCollider to trigger our character before our character falls onto the visual floor. Like this:



1. With the collider and the floor in place we need to tag our main character and the floorCollider so we can distinguish what object hit what.
   1. To do so click on “character” in the hierarchy tab>inspector tab>tag>Add Tag…>Element 0 name it floor and Element 1 name it character.

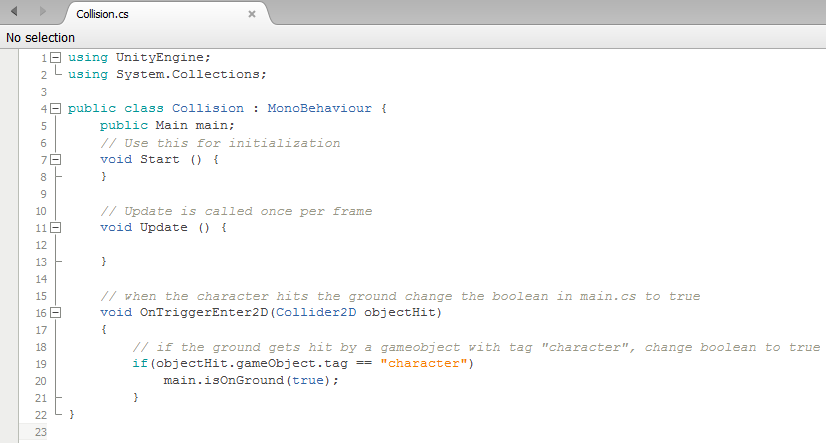


1. Click on character again from the Hierarchy tab>inspector tab>tag>character
2. Now click on the floorCollider from the Hierarchy tab>inspector tab>tag>floor
3. With the colliders all set up along with the tags we need to place a rigidbody onto our character so we can apply gravity and other forces
   1. Click on character again from the Hierarchy tab>Add Component>Rigidbody 2D
   2. Click on character again from the Hierarchy tab>Add Component>Box Collider 2D and make sure “Is Trigger is unchecked”
   3. Because our character sprite is practically a bunch of animation states the collider sometimes autofills strange size values, mine are set at 0.3 and 0.35, however it will obviously have to be altered if you use a different image.

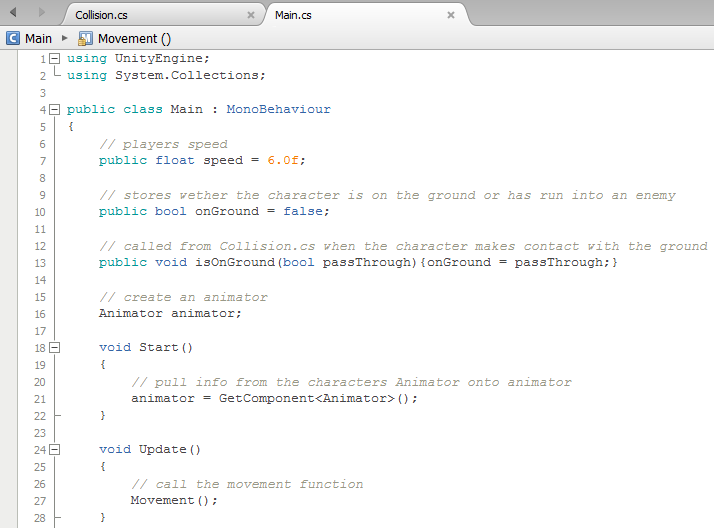
**4. Attaching Scripts**

1. Heading back to the scripts folder in the project view create 2 scripts: Collision and Main.
2. Once created attach the “Collision” script onto floorCollider and the “Main” script onto Character, simply by dragging and dropping or clicking onto character/floorCollider then clicking add component and typing the name of the script. I prefer the drag and drop method.
3. With the scripts attached copy the following code into the appropriate scripts.
   1. Comments are within the scripts to help understand how it works.

Collision.cs



Main.cs





1. With the code implemented we have to link a few variables and then our character is good to go
   1. floorCollider from the hierarchy tab>Collision (Script)>Under “Main” click the small circle to the right of it>scene tab>Character
2. Just as a reminder, make sure Character is above the ground/floor. It will fall straight down because of the rigidbody gravity and land on the ground/land which is what we want. If you place the character below the ground or inside the ground, the character won't be visible on screen because it will be constantly falling.
3. Spacebar is to jump and left and right runs in either direction, not pressing anything triggers the idle animation.