Simple 2D Top-Down Game Instructions

Dayne Dellaire 100741322

The process of making a 2D game in unity is simple and can be broken down into multiple steps. However, before we go into that it is best to understand the Unity environment you will be working in, in order to achieve the best workflow possible.

Major Tab Components for Unity

Inspector: the inspector is one of the most important tabs in unity. The inspector is the source of all the information about the currently selected object. For example if you have selected an image it will appear in the inspector tab with details about that image like size and pixels per unit.

Project: this is where the files and assets of your game is stored and managed. Everything from your images to sounds is stored in this tab. When working within the project tab it is best to keep thing concise and organized so that you can find things easily.

Hierarchy: this tab manages all the current objects that are placed in the world of your game. When creating a new unity 2D project the hierarchy will only have a camera in it, and you will have to add the rest of the game objects yourself. To add more game objects, you can right click the Hierarchy and near the bottom of the drop-down menu you will have a list of all the game objects you can use.

Tile Palette: This tab is used to help create and use Tile Palettes which can be used in level creation. The tile Palette will let you choose from a tile stored within a palette and will let you draw that tile on a tile map object. The Tile Palette has many tools to help with this, but we will mainly use the brush tool and the erase tool.

Game: Is the tab that displays how your current game would look like. This tab is purely for visual representation and cannot be directly changed.

Scene: This is the main tab to setup and drag objects around in your game. This will be your main working tab when you are trying to design a level or test out certain features of the game. It is important to also note that the scene has multiple tools in order to help you with level design and they are as follows.

- View tool (little hand) lets you use the mouse to drag around the scene without editing any of the environments

- Move tool (4 directional arrows) Used to move the selected object around the scene by dragging its 2 directional arrows, at the objects origin

- Rotate tool (2 circle arrows) Can be used to rotate the game object.
- Scale tool (Square with arrow inside) Used to scale the game object freely.
- Rect tool (Square with anchor points on the corners) Puts a box around the object with 4 anchor points which can be clicked and dragged to scale the object up and down
- Transform tool Can be used to edit every feature of the transform component such as the position, rotation and scale of the object

Every game object in Unity can have many components associated with it. These are powerful tools that can determine how a game object will interact with the environment around it.

Major Game Object Components for 2D

Transform: every game object created will ALWAYS have a transform and it cannot be removed. The transform contains the details of the objects position, rotation and scale in the game world.

Sprite Renderer: the sprite renderer is used in 2D games and when attached to an object it will render a sprite (image) on the location of the object. This also has many powerful tools like being able to select if the sprite is flipped in the X-axis (left and right) or Y-axis (up and down)

Animator: will hold a reference to an animator that will animate your game objects while they are doing certain actions or in specific states that are setup by the user beforehand

Collider2D: A collider2D is an ambiguous term for all the different types of collider2D types in unity. The collider is a shape that can collide with other colliders. For example, if you put a collider on a player and another collider on a wall, when those two collider shapes meet unity can tell the player to stop moving since a collision has happened.

RigidBody2D: Is used for handling the physics of the game the rigid body has 3 different types associated with it. The first being. Static which is made for a game object that wont be changing position. The second is Kinematic for a game object that will do some movement and not be able to have any forces acted upon it, and the last being Dynamic which is used for more dynamic objects that will be moving and be able to have forces put upon it.

Assets

Firstly before we can construct our game it is best to find some assets. When working in a 2D environment the images are referred to as sprites. A sprite sheet is a collection of frames that when put together will resemble an animation for that sprite. A tile set image contains a series of images that will often make up an environment. For example, the Ground Tile set contained in the Assets/Sprites/Forest folder is a tile set that contains many 16x16 sprites that can be used to construct a level.

If an imported sprite is 2D it is important to enable / disable some options first before they can be used.

1. Import a sprite by dragging it into the project tab.
2. Click on the sprite in the project tab and view it in the inspector

If you are working with pixel art, follow steps 3-5

3. Change the pixels per unit to the resolution of the sprite. The character in the game is 16x16 pixels so the Pixels per unit is also 16x16. In order to make the art look similar make sure that the PPU (pixels per unit) is the same for each imported sprite
4. Click the advanced drop-down menu and change the filter mode to point.
5. Lastly change the compression to none

If you are working with multiple sprites in one, image follow steps 6-9

6. Change the Sprite Mode to Multiple and click Apply at the bottom.
7. Click sprite editor to open the sprite editor tab. This tab will let us slice our sprite sheet so we can extract the multiple images from it.
8. At the top of the Sprite editor click slice and change the type to grid by cell size and change the cell size to the resolution of each individual sprite.
9. Click Slice and then exit the sprite editor.

10. Click Apply at the bottom of the inspector and you're done!

To test the sprite, you can create an Empty game object and name it Player. Inspect the player object and click add component from here search Sprite Render. Select it and add it to the object. In the newly added sprite renderer component. Drag the sprite you want to render from the project tab into the sprite category in the sprite renderer, and you should see the sprite appear. Also add a RigidBody2D component to the player object for later physics calculations and movement. However, since this is a 2D game set the type to Dynamic and Gravity Scale to 0

User Input

The first step to any game is how the user is going to interact with it. For a top down 2D game a popular set of basic inputs is using the WASD keys for movement and having extra buttons for extra actions such as mouse left click for attack. In order to do this lets first create a new script to handle some movement.

1. Inspect the player game object we created in the last step.
2. Click add component in the spectator tab.
3. Type “Movement” and create a new C# script.

From here the movement script will be created in the root asset directory and now we can get to scripting. While these instructions wont go in depth on scripting there are a few things to note that are specific to Unity game engine.

Update: This method is called every frame and can vary depending on the performance of the users computer or what frame rate the game is locked to so it is best to avoid any physics calculations in this method
FixedUpdate: This method is called every time Unity does a new physics calculation and this rate can be changed in the system settings but for now, we can leave it as default.
Start: This method is called on the first frame when a script is enabled it will run once and only once So it is best to assign starting data here

As a best practice you must remember that any sort of calculations should most of the time be put in the fixed Update method where as anything revolving player input and animation will be created in the update method. Knowing this let’s set up some player input. In the newly create movement script lets write some code. By default, unity will add the Start and Update methods to the script. In the script under the class definition create 2 new global variables. One private vector2 called moveDirection and a public float called speed. A vector2 is a simple data type which carries a x and a y direction and will help determine the movement direction of the player.

In the Update function add the code moveDirection.x = Input.GetAxisRaw(“Horizontal”). This line of code will get if the user is pressing the A / Left arrow key or the D / Right arrow key and return 1 / -1 if a user is pressing a key and 0 if a user is not. The same method can be applied for the y direction moveDirection.y = Input.GetAxisRaw(“Vertical”). Now that the inputs are retrieved. Last thing to do is move the object the script is attached to. To do this we will use the rigidBody2D component of the player.

To edit components from script first we must create a new global variable. This one being a reference to the Rigid body 2D component. Under the speed variable type public Rigidbody2D rb. Finally putting it all together so that we can move the player object create a new method called FixedUpdate in this method we write rb.velocity(moveDirection.normalizied * speed). Finally inspect the player object now and under the movement script component you should see 2 empty fields. Set the speed to whatever you like and drag the rigidbody2d component into the rigidbody field.

There you have it the user should be able to move the character around the screen using the WASD and Arrow keys.

Level Creation

Level creation can be done in a few ways. There is no perfect way to do level creation. However, one of the most popular ways to create a level is by using tile maps. Tile maps are a powerful tool that allows you to draw tiles you have created into the scene in order to construct levels very quickly with minimal effort. Used the Ground Tileset contained in the Assets/Sprites/Forest folder lets create a Tile map.

1. In the Hierarchy tab right click to start creating a new object. Proceed as follows
   2D Object > Tile map > Rectangular.
2. You will notice that it creates a Tilemap that has a parent which is the grid. This isn't important. However, make sure to rename the Tilemap to something memorable so that you will be able to access it later.
3. Click Window > 2D > Tile Palette, this will open the Tile Palette tab which will let you create a Tile Palette. Drag the tile Palette over to the right next to the inspector
4. Take your Tilemap sprite from your project tab and drag it into the Tile Palette
5. The Tile Palette should prompt you to choose a destination to save the Tiles and the Tile Palette. Choose the destination and Unity will automatically create the tiles for you from the sprite image.

Once the tile palette is setup you are free to paint. Firstly, make sure the Active Tilemap option in the tile palette is set to the new Tilemap you created in step 1. Then select the brush tool or press the key B, and choose a tile from the tile palette and get to painting in the scene tab. If you make a mistake and want to undo it press CTRL+Z or press the D key to select the eraser tool and remove the mistake.

You can also create multiple tile maps in the same grid in order to layer tiles on top of one another. However if you are doing this the case of having a sprite appear on top of one another can be tricky to manage especially as the project continues to grow. When inspecting the Tilemap you can change the Order in Layer option in the Tilemap Renderer component in order to change which tile will appear on top of the other.

Note: It is better to use Sorting layers in order to help maintain readability and keep track of layers however I won't explain that

Sprite Animation (Via Sprite Sheets)

Sprite Animation is a big task. But unity makes it simple. Here are the steps breaking down how to animate a simple sprite using a sprite sheet.

1. Open both: Window > Animation > Animation
   Window > Animation > Animator
2. If the Animation tab appears greyed out make sure you have an object selected in the Hierarchy that you want to animate. If you do not for now create an Empty game object that can represent a player.
3. Click Create in the animation tab and now you can start creating an animation.
4. Drag sprites into the animation tab and they will create keyframes
5. The Keyframes can be dragged around the timeline and the separation will determine the time between when each frame is played.

This Technique can be repeated and used to animate many objects inside of Unity. In the example game I created I made an animation for 5 different directions the player could walk. The animations become a key component of the game and will help add visual flair. In order to create a game of your own make sure you at least have a walking animation and maybe a idle animation.

Setting up the Animator
To continue the example, inspect the player object, click add component and search animator. Add the component to player object. When inspecting the Player object you should see that in the animator component it has automatically assigned the Player controller to the Controller section. Now we can get ready to animate.

Animation States

Go back to the Animator tab and you should have a basic layout for an animator. In the Animator there are 3 base states those being:

- Entry: will go into this state when first going into the animation
- Any State: Can transition to this state from any other state.
- Exit: return to the entry state

From these basic states you should also see your newly created animation as well. Newly created animation states will appear when you create a new animation for the object. However if it doesn't appear you can drag your animation from the project tab into the Animator tab.

Parameters

In order to go in between animations, we need to setup some parameters. You can view parameters in the left of the animator. Create 2 float parameters and name them moveDirX and moveDirY, we will update these parameters in the movement script. To do that we need a reference to the animator component in the movement script, once we have a reference to that we can use the animator.setBool("moveDirX", moveDirection.x) function in the FixedUpdate method to Update the parameters.

Transitions

You may also create animation state transitions so that when your game is running the animator will dynamically transition between states so that your game object will be fully animated. When you create the first animation for any object you should see a transition from the basic entry state to your state. If you want to add more animations, you will have to setup more animations and bring them into the animation state tab.

Blend Tree

To fully animate a character with 4 walk directions right click an empty space in the animator tab and choose from new blend tree. Double click on the newly created blend tree to access it. We will use a blend tree combined with the inputs setup before to create the animator clean. The blend tree is a powerful tool which will help us set the animation state we want. When in the blend tree change the type to a Simple 2D directional and change the parameters to the MoveDirX and MoveDirY that we created before. Then create 4 motion fields. Each motion field has an X and Y component and we want to set each one of these so that they are in the 4 cardinal directions. Once that is complete set each motion field with an animation.

To put it all together from the entry state create a new transition to the blend tree and then the work is done. The character should change animation when the user provides input.

Collisions

Finally, Collisions. Unity makes collisions simple especially when working in a 2D environment. Unity has a few types of 2D colliders the main ones being a Box Collider and a Circle Collider. The only real difference between the collider components in the shape in which they take.

Collider2D

Add a box collider 2d component to the player and to edit its size and shape click the edit collider button in the inspector and you can drag its anchor points in the scene tab. Since the player has a rigidbody 2d component already the physics system in unity will handle the collisions.

Tilemap Collider

However, the player needs other colliders to collide with. There is many ways to do this however when we are using tile maps one of the most effective ways to do this is to use a Tilemap collider 2d. A Tilemap collider is a powerful tool that will automatically generate a collider for Tilemap tiles. You can add this component to your Tilemap you wish for the player to collide with for example another Tilemap named walls.

There it is if all the steps are followed correctly then you should have a working game with basic input, sprite animation, and collisions. You also have a Tilemap created for easy level creation.