DODGER GAME:

Dodger is a small, simplistic game that doesn’t take much logic to program and can be customized almost endlessly. Dodger only has a few key elements of Gameplay:

1. Player Input

2. Player/Enemy Bounds

3. Random Location Generation

4. Collision

5. Lose Condition

Using these five elements we will create a simple Dodger clone. This tutorial assumes you have installed Unity.

Adding the Assets

First, you will want to download the images from the GitHub page (or right-click and save image as…).

To create folders in Unity you need to Right-Click inside of the Assets Pane and select Create -> Folder. Name your new folder Images, and create 2 more folders named Prefabs and Scripts.

After creating your folders you will be able to drag the images you downloaded from the Github page into the Images folder. Select both of the images inside of your Images folder and set their Pixels Per Unit value to 32 and click Apply

Creating the Player

To add player input we will first have to create a Player game object. You can create a Player game object by dragging the image of the green smiling face into your hierarchy pane. Set the name of the game object to Player and set its Y Position to -3.5

Modifying Our Camera

If we left the camera like this our sprites would have to use decimal numbers in order to stay just inside the screen like we want. Unity allows us to change the size of the main camera and its background color. Select the camera and click the background color. Set the color to a beautiful shade of blue (R: 0 G : 223 B: 255).

Getting Player Input

We’ll start off our programming by getting player input. In order to get player input we will need to create a new C# Script and name it Player Controller.

Add the Player Controller script to the Player game object in the hierarchy by dragging it from the Assets pane onto the Player game object. If you run the game and try to control the Player using the arrow keys you will see the player move on screen.

The Enemy

Next we will need to create the enemy. In order to create the enemy you will need to drag the angry green block into the hierarchy pane. Go ahead and name the game object Enemy. Click under the game object’s name where it says Untagged then click “Add Tag”. Create a tag named “Enemy”, and give the Enemy game object this tag. Select the Player game object and give the it the tag “Player”.

Now that we have the tags set which will tell the difference between the Player and the Enemy, we can start coding the enemy’s behavior. Open the Scripts folder in the Assets pane and create a new C# script named “EnemyController”. Add the EnemyController script to the Enemy game object. Open the script in your IDE and type this into your script:

Adding Collision

Inside of Unity select the Player game object. In the Player game object’s Inspector pane click the Add Component button, search for “Circle Collider 2D”, and add it to the Player game object. Do the same again, except search for “Rigidbody 2D” and add it to the Player game object. On the Rigidbody 2D set the values to the same as the image below.

Next select the Enemy game object, and inside of its Inspector pane click on the Add Component button. Search for “Box Collider 2D” and add it to the Enemy game object. Check “Is Trigger” inside of the menu for the Enemy game object’s Box Collider 2D.

Open the EnemyController script inside of your IDE and under the Update method type:

Randomly Spawning the Enemy

Inside of Unity drag the Enemy game object in to the Prefabs folder inside of your Assets pane. Delete the Enemy that is inside of the hierarchy since we will be randomly generating the location of all of the spawned enemies. Right click inside of the hierarchy pane and select “Create Empty”. Name the empty game object “EnemySpawner” and sets its Y position to 6. Inside of the scripts folder create a new C# script named “EnemySpawnerController” and open it inside of your IDE.

Adding Basic UI

Inside of the hierarchy pane, right-click and select “Create Empty”. Set the new game object’s name to “UIManager”. In your scripts folder create a new C# script and name it “UIManager”. Open the script in your IDE and type:

Finishing Up

Create a “Scenes” folder in the Assets pane. Save our current scene as “MainLevel” and it to the scenes folder. From the top menu select File -> Build Settings and drag the MainLevel scene from the Assets pane into the Build Settings menu or hit “Add Current” in the menu. Finally, build the game to either PC, Mac, or Web and have fun playing

Conclusion

This short but to-the-point tutorial went through the exact steps of creating a clone of the popular Dodger game in Unity. Hopefully, you liked it and are craving for more Unity content – it’s coming soon, we promise! Feedback? Comments? Leave them below!