Contents

[Project Folder Setup 2](#_Toc495404084)

[Adding Sprites 2](#_Toc495404085)

[Scene Layout 3](#_Toc495404086)

[Object Naming 3](#_Toc495404087)

[Ball Object 4](#_Toc495404088)

[Circle Collider 4](#_Toc495404089)

[Physics Material 2D 5](#_Toc495404090)

[Rigid Body 2D 6](#_Toc495404091)

[Barriers and Goals 7](#_Toc495404092)

[Box Collider 7](#_Toc495404093)

[AI Paddle 7](#_Toc495404094)

[Box Collider 7](#_Toc495404095)

[Player Paddle 8](#_Toc495404096)

[Box Collider 8](#_Toc495404097)

[Rigid Body 2D 8](#_Toc495404098)

[Scripting 9](#_Toc495404099)

[Creating Scripts 9](#_Toc495404100)

[Assigning Scripts 9](#_Toc495404101)

[Player Controller Code 10](#_Toc495404102)

[Variables 10](#_Toc495404103)

[Update 10](#_Toc495404104)

[AI Controller Code 11](#_Toc495404105)

[Variables 11](#_Toc495404106)

[Update 11](#_Toc495404107)

[Setting BallToFollow 11](#_Toc495404108)

[Ball Controller 12](#_Toc495404109)

[Variables 12](#_Toc495404110)

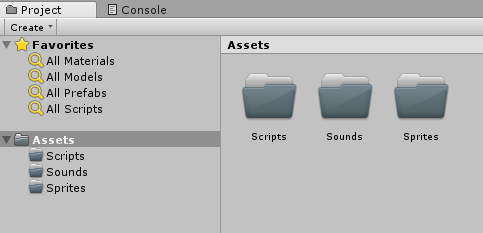
[ResetBall 12](#_Toc495404111)

[Start 12](#_Toc495404112)

[OnCollisionEnter2D 12](#_Toc495404113)

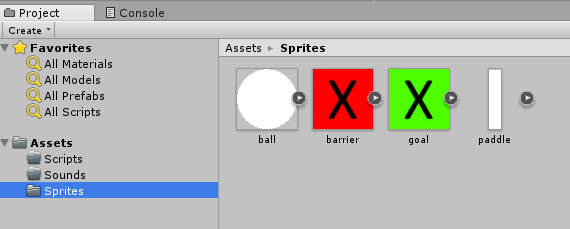
## Project Folder Setup

* Create three new folders in the Unity Project
  + Scripts
  + Sounds
  + Sprites



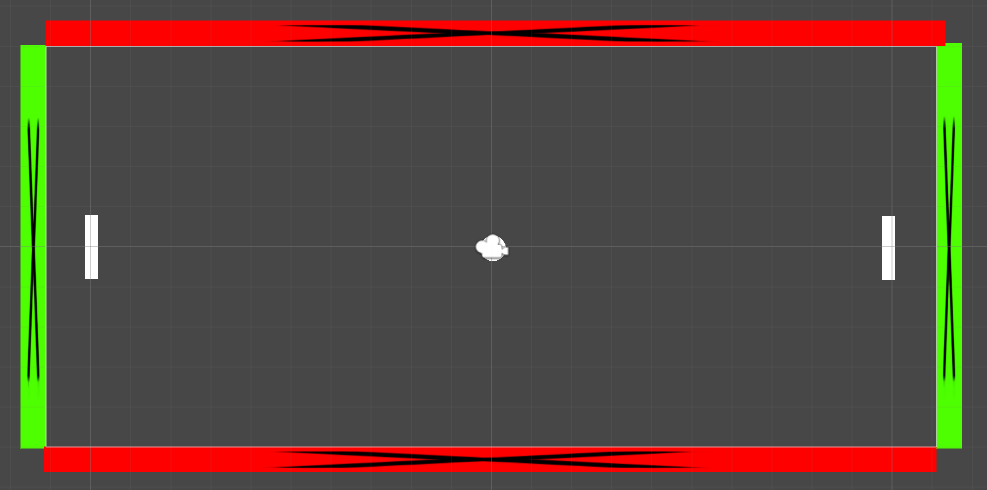
## Adding Sprites

* Download the required sprites from Moodle
* Drag and Drop the 4 sprites into the Sprites folder in Unity



## Scene Layout

* Drag and Drop the sprites into the scene
* Layout your scene as shown in the image below
  + One ball in the centre
  + Two paddles on either side
  + Two goals (one behind each paddle)
  + Two barriers (one on top and one on bottom)
* Click the play button to ensure that the goals and barriers are not visible when playing the game
  + If they are visible then adjust their position until they no longer are



## Object Naming

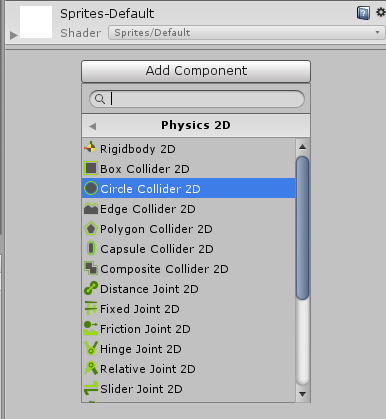
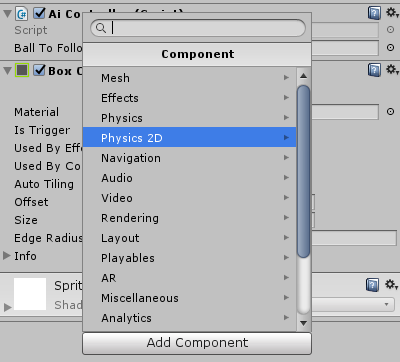
* Rename the object you have added to the scene
  + Left paddle = player
  + Right paddle = ai
  + Left goal = leftgoal
  + Right goal = rightgoal
* The names should match what is shown below
  + Ignore the order in which the appear in the list



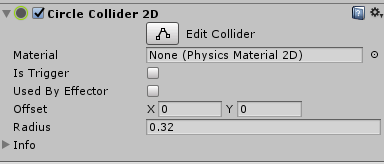
## Ball Object

### Circle Collider

* Select the ball object in the scene
* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Circle Collider



* The circle collider component will now be added to the ball

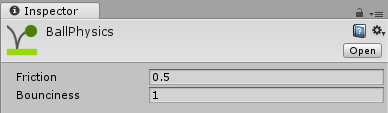


### Physics Material 2D

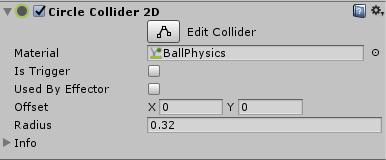
* Right Click in the assets window of the project
* Select Create -> Physics Material 2D
  + Name it BallPhysics



* Select the newly created asset and notice that properties that are displayed in the inspector
  + Modify them to match those shown below
  + Friction = 0.5
  + Bounciness = 1

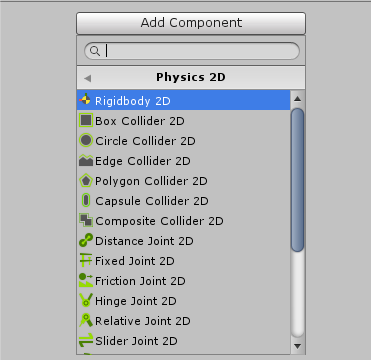
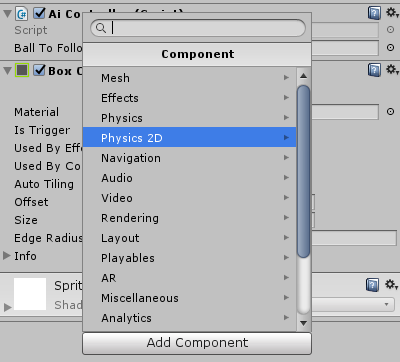


* Select the ball object in the scene
  + Drag and Drop the new BallPhysics onto the Material box of the Circle Collider
* The ball will now use the physics material to bounce in the game

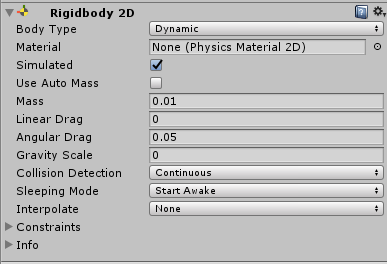


## Rigid Body 2D

* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Rigid Body 2D



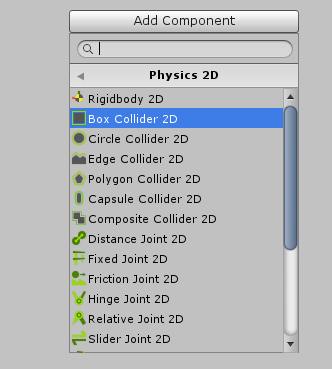
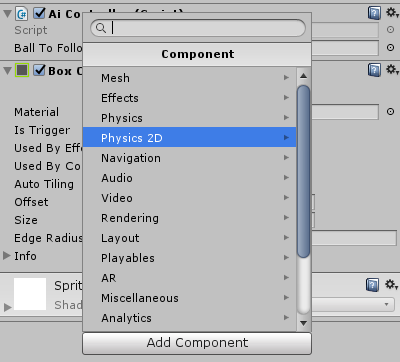
* We need to modify the default values of the rigid body for the ball to work correctly
  + Set the **Mass** to 0.01
  + Set the **Gravity** Scale to 0
  + Set **Collision Detection** to Continuous



## Barriers and Goals

### Box Collider

* Select the barrier object in the scene
* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Box Collider

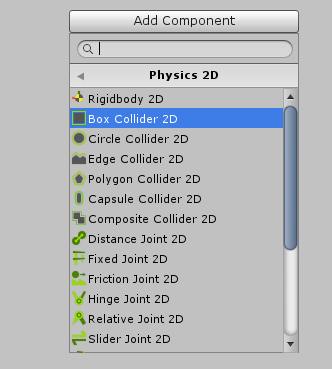
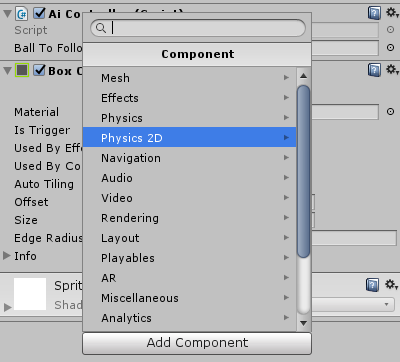


* Repeat this for the remaining barrier and the two goal objects

## AI Paddle

### Box Collider

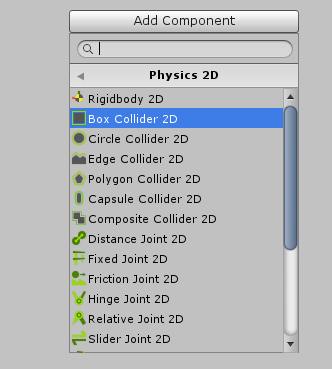
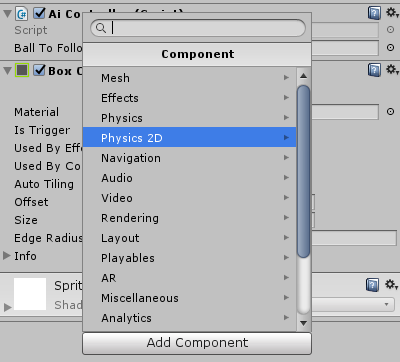
* Select the ai paddle object in the scene
* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Box Collider



## Player Paddle

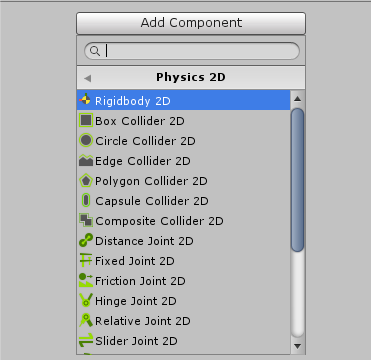
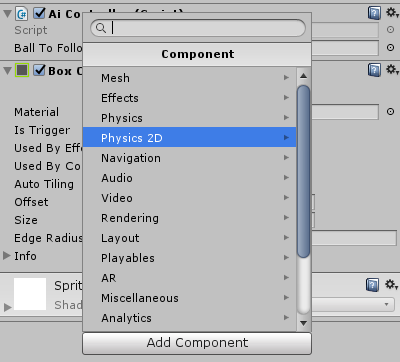
### Box Collider

* Select the player paddle object in the scene
* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Box Collider

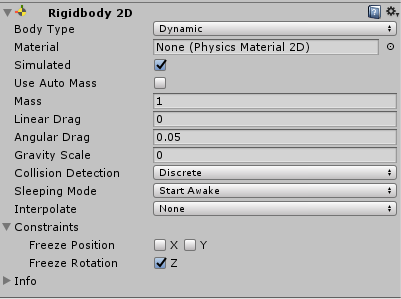


### Rigid Body 2D

* In the inspector you will see an Add Component button.
* Click this button
  + Select Physics 2D -> Rigid Body 2D



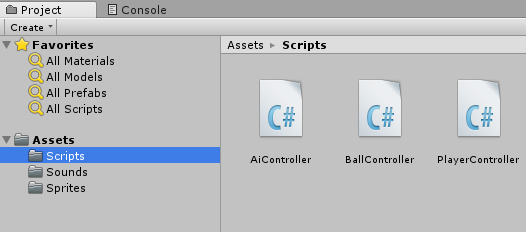
* We need to modify the default values of the rigid body for the ball to work correctly
* Expand the Constraints Section
  + Check the Freeze Rotation Z box



## Scripting

### Creating Scripts

* With the Scripts folder selected
  + Create three new C# Scripts
  + Right Click -> Create -> C# Script
* Create Three new scripts
  + AiController
  + BallController
  + PlayerController



### Assigning Scripts

* To assign a scrip we need only drag and drop it on the inspector window for a select object
  + Select the **ball** object in the scene and drag and drop **BallController** into the inspector window
  + Select the **ai** object in the scene and drag and drop **AiController** into the inspector window
  + Select the **player** object in the scene and drag and drop **PlayerController** into the inspector window

### Player Controller Code

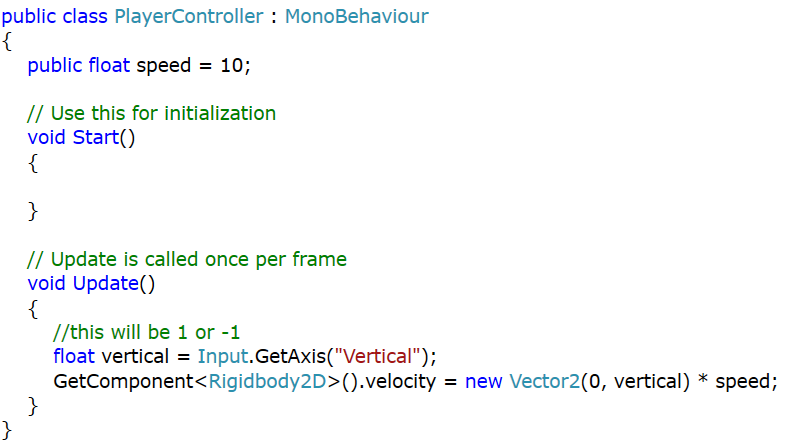
* Double click the **PlayerController** script and wait for Visual Studio to open

### Variables

* Add a single **speed** variable of type **float** with a default value of **10**
  + The public keyword means that we can edit this value directly in the inspector in the unity editor

### Update

* We need to move the paddle when the player presses the Up/Down arrows
* Unity provides a built in way to get determine whether the Up or Down button has been pressed
  + **Input.GetAxis(“vertical”)** will return one of three values
  + 0 no button was pressed
  + 1 Up button was pressed
  + -1 Down button was pressed
* Multiplying our speed variable by retuned value will tell our paddle which vertical direction it should move in
* To move our paddle we need use **velocity** of the **RigidBody2D** we have attached
  + We first must get the **RigidBody2D** component
  + We hen access its **velocity** variable
  + We set the **velocity** to be a new variable (**Vector2** is just an X,Y value)
  + The Vector2 will be our vertical movement direction \* speed
* Save the code
  + Save Icon or Ctrl+S
* Go back to Unity and test the code works
  + Click the play button on the top of the editor



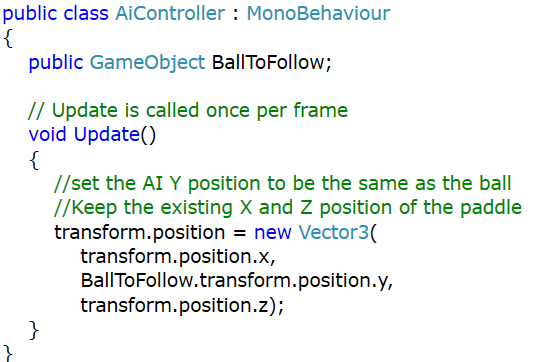
### AI Controller Code

### Variables

* Create a new variable of type **GameObject** and call it **BallToFollow**
  + Later we have to fill this variable with the ball placed in the scene

### Update

* In the Update we want to make the AI follow the Y position of the ball
  + We still need to maintain the X and Z position of the paddle
  + Note that Z is used for depth and is not of any use to us today
* Set the **transform.position** to be a new **Vector3** (just and X,Y,Z position)
  + Set the **X coordinate** to be the current X coordinate of the **paddle**
  + Set the **Y coordinate** to be the current Y coordinate of the **Ball**
  + Set the **Z coordinate** to be the current Z coordinate of the **paddle**
* Save the code
  + Save Icon or Ctrl+S
* Go back to Unity



### Setting BallToFollow

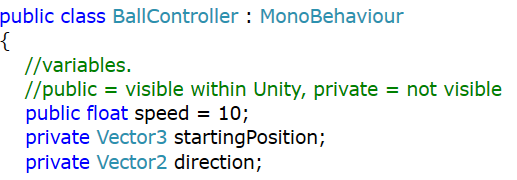
* Select the ai paddle in the scene
* Drag the ball from the scene onto the Ball To Follow box that now appears on the AiController script
* Click the play button on the top of the editor



## Ball Controller

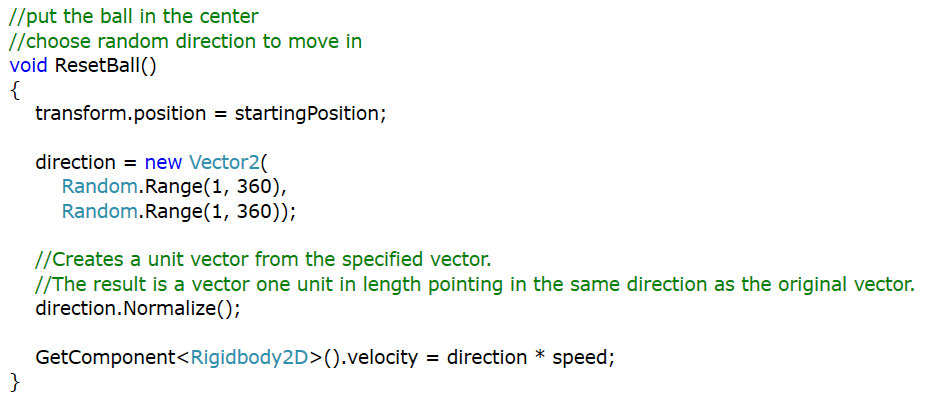
### Variables

* We need three variables for our ball to work correctly
* A **speed** variable with a default value of **10**
* A **startingPosition** variable
* A **direction** variable



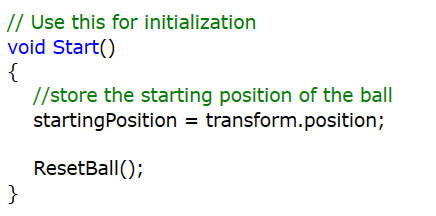
### ResetBall

* We need to create our first C# **method** called **ResetBall**
* When this method is called
  + Set the ball **position** to be value stored in **startingPosition**
  + Set the **direction** to be random **direction**
    - X and Y and both **random** angles between 1 and 360
  + **Normalize** the **direction**
  + Set the **velocity** of the ball to be the direction \* speed
    - This will **move** the ball in the specified **direction** by the specified **speed**



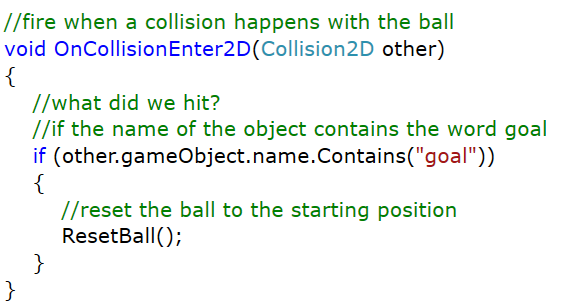
### Start

* In the **start** method
  + Set the **startPosition** variable to the **position** of the ball when the game starts
  + Call **ResetBall** method
    - Picks a random direction and starts the ball moving in that direction



### OnCollisionEnter2D

* When the ball collides with a goal we want it to go back to the centre and start moving in a random direction
* Add the **OnCollisionEnter2D** method
* **Unity** will call this method when the ball collides with another object
  + Note the **Collision2D** argument that is passed into this method
  + This contains the other object the ball hit
* We can check if the **name** of the other **object** **contains** the word **goal**
  + If it does, then call **ResetBall**
  + Otherwise do nothing
* Save the code
  + Save Icon or Ctrl+S
* Go back to Unity and test the code works
  + Click the play button on the top of the editor



## Full Ball Controller Code

