PINBALL

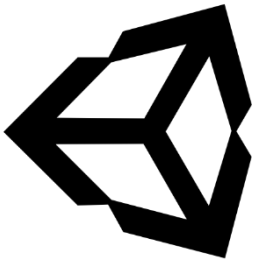
We're going to be making Pinball! It's a fairly simple game to play, but recreating it is a different challenge. A ball is launched from one side into a playing area with bumpers and walls. Gravity pulls the ball downwards towards a pit, and the player must keep it in the play area by controlling flippers that kick it upwards. We're going to make a simplified version of this using a program called Unity.

Unity is a *game engine* which is free to download. We're going to get familiar with Unity by designing our own take on a pinball level. Everyone's level will be slightly different!

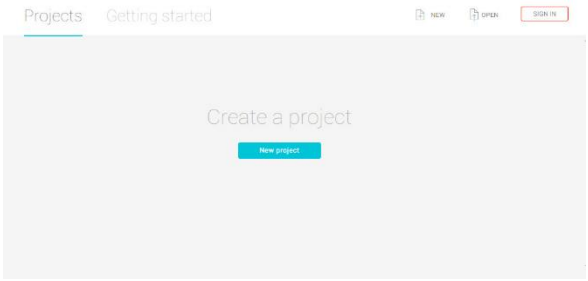
TABLE OF CONTENTS

OPENING UNITY

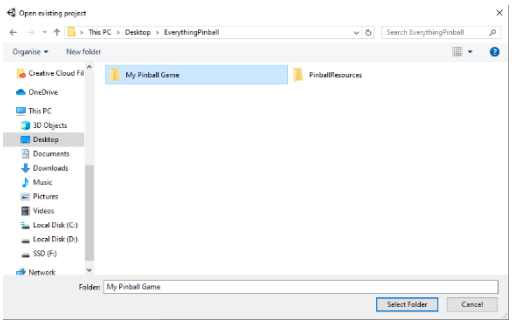
1. Double click the Unity icon on the desktop. If the icon isn't there, you may need to search for it from your desktop.



2. Once Unity opens, you should get a window asking you to sign in. Click on "Work Offline".



3. Click on Open. Locate the project folder and Click "Select Project"



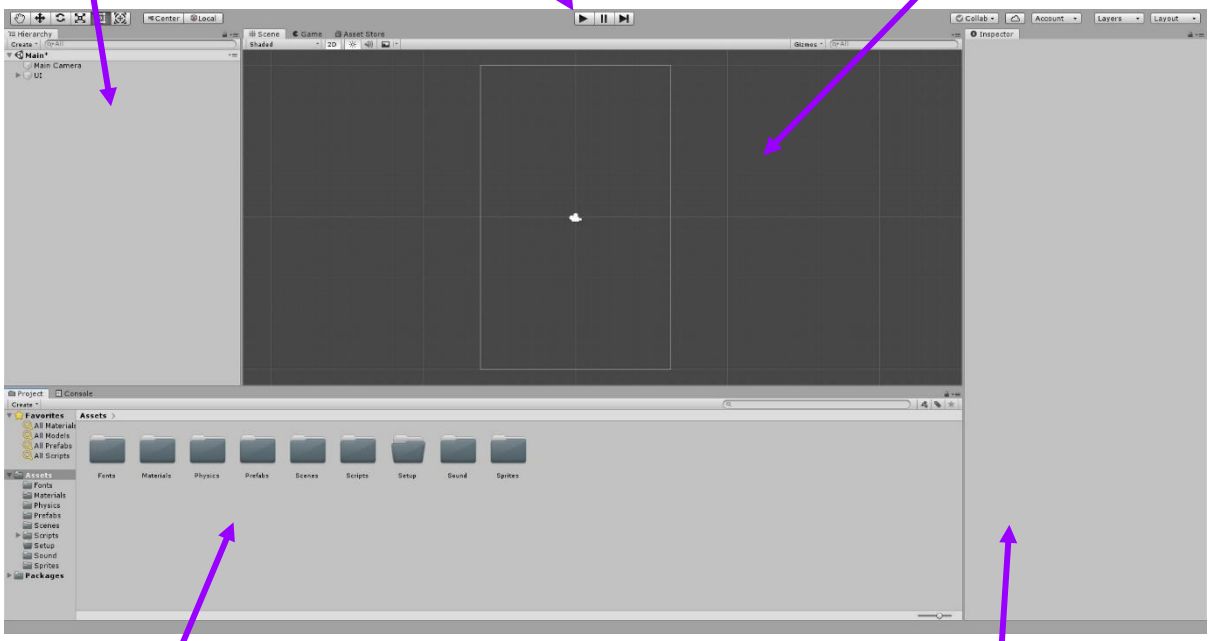
Once the project is set up, it will bring you to the main Unity window. Yours may be dark rather than light grey.

Let's break down the important parts of what we see here:

Hierarchy
A list of every **object** you have in the game scene

Preview Button
Lets you play the game.
Remember to exit preview mode before making changes!

Scene View
A view of the game world.



Project
Every file that makes up your game goes here. It works like an ordinary folder on your computer. Its main folder is called **Assets**.

Inspector
Every game object is made out of **components**. The inspector lets you see these details when something is selected.

You can use the scroll wheel to zoom in and out of the Scene View.

Don't worry if it doesn't all make sense! Learning a new program can be tricky, but everything you need will be explained along the way.

SCENES

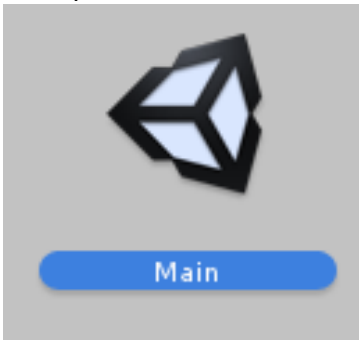
Before we start, there are a few more things we need to do.

A **scene** in unity is a game world - it contains every object that has been placed. Scenes are a file that need to be saved like in any other program.

Assets is where all of the files that make up the game go - this includes the scene files. Since we will have a lot of files, we need to keep things organised with folders.

Fortunately, we already have a basic scene for you to being creating, and located in the *Scenes* folder.

1. Double click the Scenes folder.
2. Open the double click the scene called “Main”



Remember to save often! CTRL + S is the keyboard shortcut to save.

OBJECTS

An **object** is a 'thing' in the world. Every **Object** is listed in the **Hierarchy** window on the left. Notice that there are a few objects already in the hierarchy - the Main Camera and UI.
Let's think what objects we will need for this game:

- **Ball** – They fall, roll and bump into things
- **Boundaries** - so the ball can't go off screen
- **Flippers** – controlled by the player to prevent the ball from falling

Anything else? *Hint: There is more.*

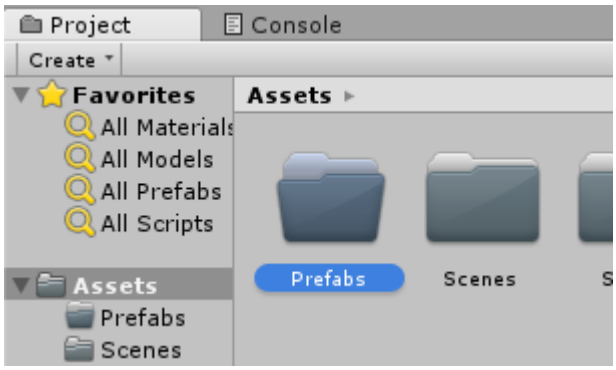
Fortunately, we have prepared these objects for you! Your job is to use these objects to design your own pinball level. We are going create objects from **Prefabs** in the Project view.

PREFABS: SETUP

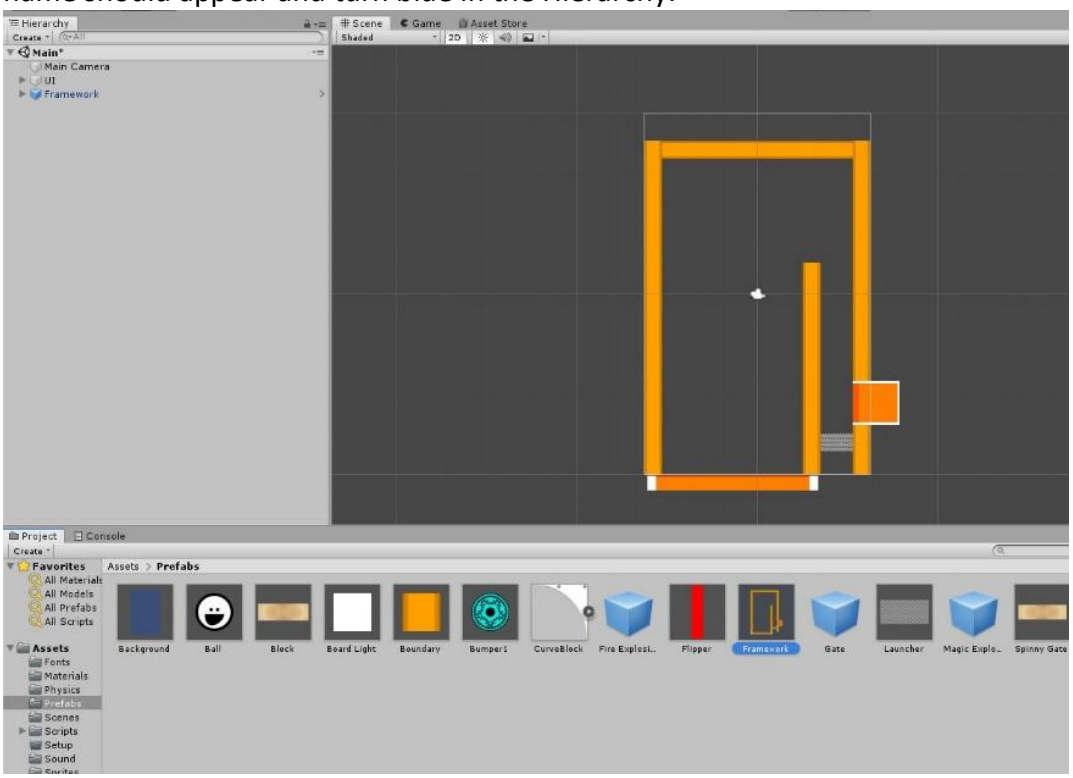
Prefabs are another word for *templates*. They are a blueprint that you can copy from as many times as you like and make the same object each time. You might make a prefab for a bullet in a shooter game, or a turtle enemy in Super Mario, because they are repeated often in their games.

Let's get one of our prefabs out now.

1. In the Project view, open the folder named "Prefabs"



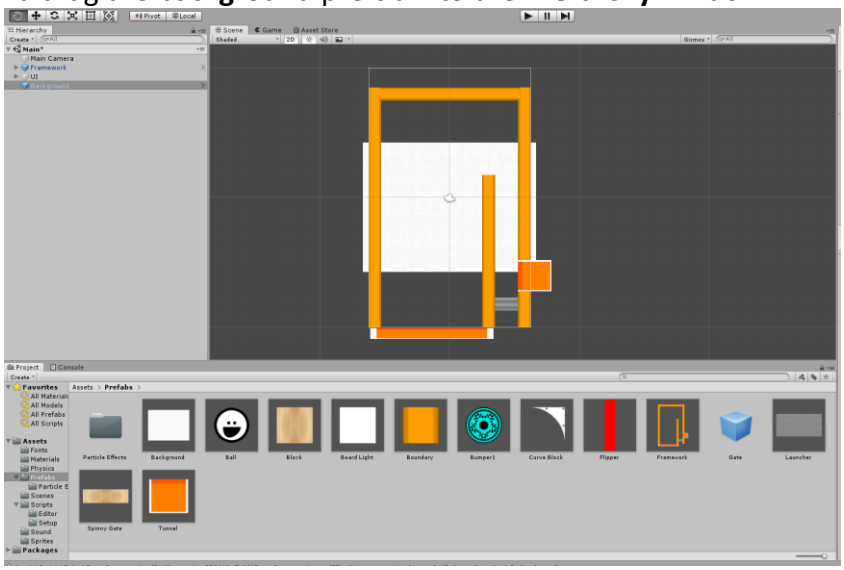
2. Click and drag the **Framework** prefab into the **Hierarchy** window. You will notice its name should appear and turn blue in the Hierarchy.



Nice! The framework object keeps the ball contained in an area and has a launcher system that helps us push the ball into the playing area.

Next, let's add a background behind the framework.

- Click and drag the **background** prefab into the **Hierarchy** window.



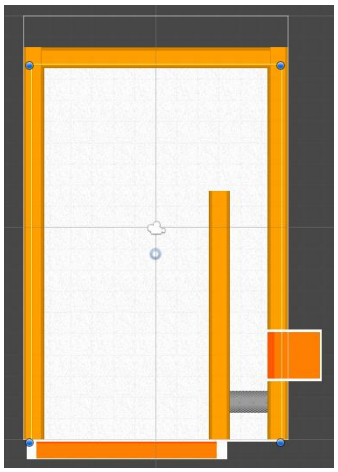
You will notice that it's not quite the right shape, but we can change that real quick.

Above the hierarchy you will find a row of different tools that help us manipulate objects in our scene. Today we are going to be using the **Rect Tool**. Click the icon as indicated below:

- Click the **background** object in the hierarchy.

You should now see that the background in the scene view has 4 blue handles in its corners. You could use these handles to resize, rotate and stretch it, much like an image in a word document.

- Use the Rect Tool to reshape the background image so that it covers the inside of the framework.



Now that it's in place, why not give it some colour?

COMPONENTS

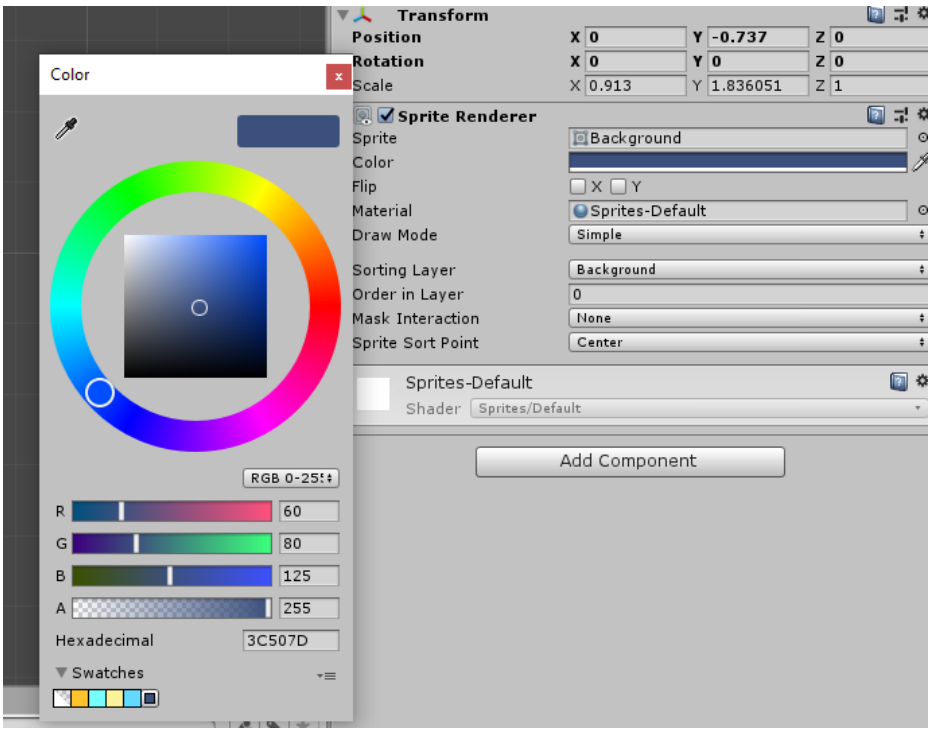
Game objects are made up of different **components**. Every component has a different purpose, and together they work to make an object behave the way you want it to.
Click on the **background** object and have a look at the Inspector window - it shows two components that are attached to the object.

Transform - This component handles the position, rotation and scale of an object in the world.

Sprite Renderer - This component makes the sprite visible on the screen.
Colour is listed as a **property** of the Sprite Renderer component.

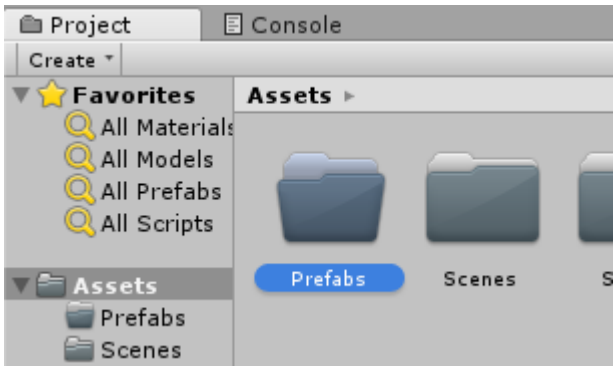
Let's go ahead and modify the colour of our background.

- Click the white *Color* bar in the Sprite Renderer component.
- Change the colour of the background using the colour picker window that opened.

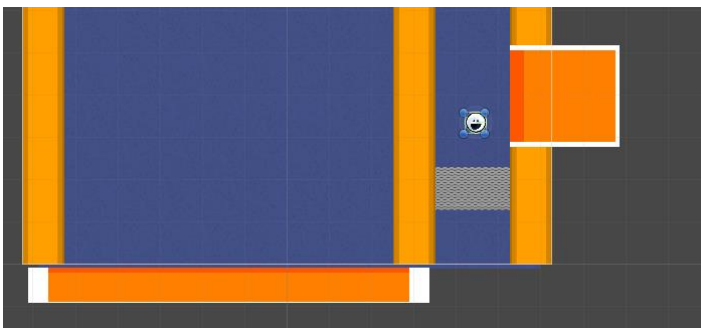


Awesome! Let's get a few more essential objects in our game, like the ball.

- In the Project view, open the folder named "Prefabs"



- Click and drag the **ball** prefab into the Hierarchy window.
- Using the **Rect Tool**, move the ball to the right column just above the launcher as follows:

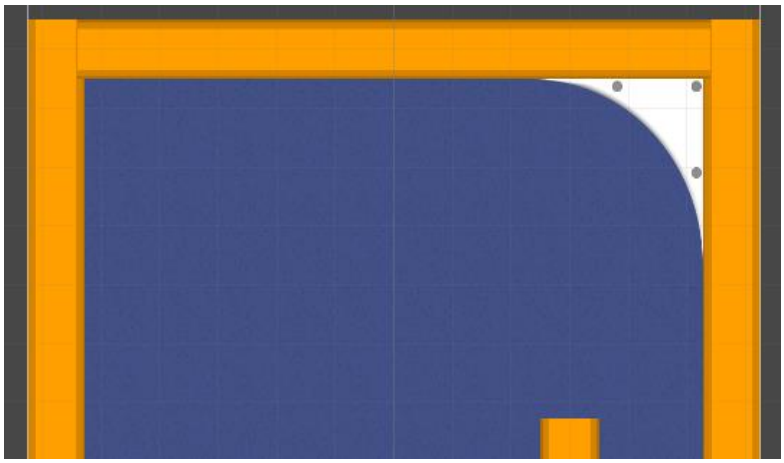


Let's click the **Play** button to preview the game and see what happens!

- While in preview, hold and release space to use the **launcher**.

Awesome! The ball falls and the launcher pushes the ball upwards! But we have a problem. the ball won't move outside of the column. We need an object that lets the ball roll into the playing area.

- Click and drag the **Curve Block** prefab into the Scene window.
- Using the **Rect Tool**, move and stretch the Curve Block object so that it covers the top-right corner of the framework as follows:



- Click the play button to make sure the ball exits the column and enters the play area.

IMPORTANT: Make sure to **exit preview mode** when making changes. If you make changes during preview mode, they will be deleted!

Awesome! Now, let's get start putting objects in our play area!

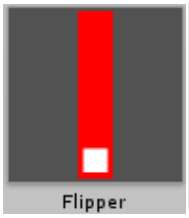
The list below outlines some other prefabs that you can drag and drop into your scene. Some of these objects have an extra **component** in its inspector that you can manipulate. These components are outlined in the list as well.

Feel free to pick and choose which prefabs you would like to copy into your scene!

Don't forget about using the Rect Tool!

PREFAB LIST

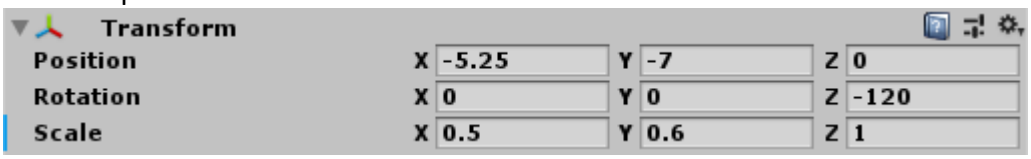
FLIPPERS



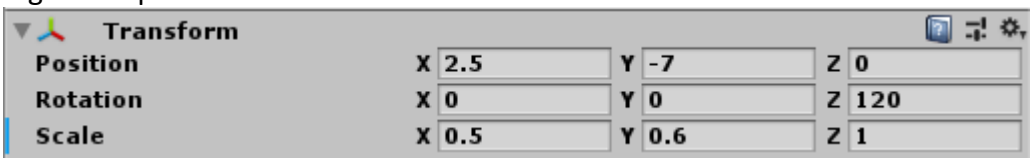
Rotates when the player presses a specified button. Usually designed to hit the ball upwards and keep it from falling.

Below is a recommendation of the Transform values for two bumpers. After dragging two copies of the flipper prefab, copy the values of the Transform component for the following:

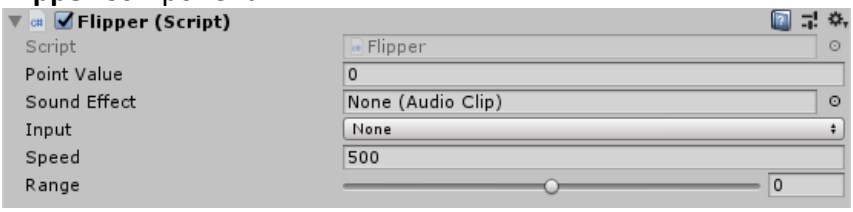
Left Bumper:



Right Bumper:



Flipper Component:

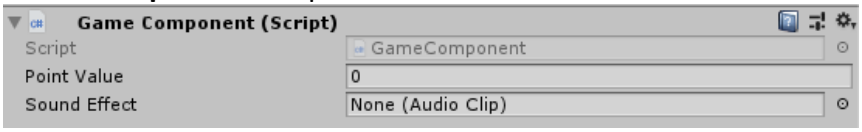


- Point Value: Number of points awarded when a ball collides with it
- Sound Effect: The sound played when a ball collides with it. Click the circle to the right of this to view all sounds.
- Input: The keyboard button you hold to activate the flipper.
- Speed – How fast the flipper will rotate.
- Range – The angle in degrees that the flipper will rotate to when activated.

BUMPERS

A bouncy circle that usually awards the player some amount of points when hit.

Game Component component:

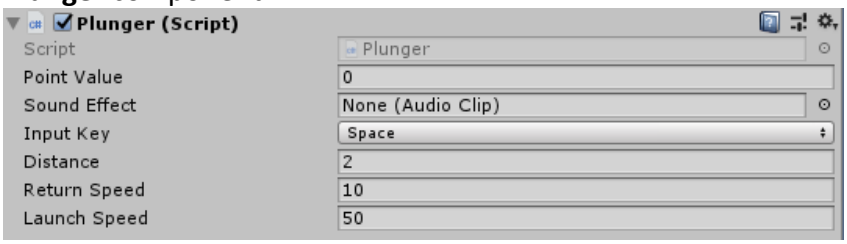


- Point Value – Number of points awarded when a ball collides with it
- Sound Effect – The sound played when a ball collides with it. Click the circle to the right of this to view all sounds.

LAUNCHER

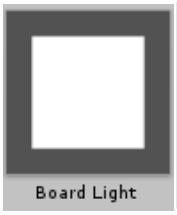
A rectangular piece that moves back while holding a keyboard button, then jolts forward when you release it. Found in the launching column, but you can also add more in the play area.

Plunger component:



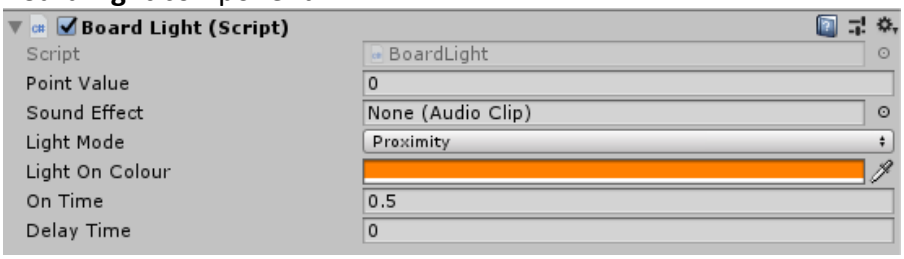
- Point Value – Number of points awarded when a ball collides with it
- Sound Effect – The sound played when a ball collides with it. Click the circle to the right of this to view all sounds.
- Input Key – Button on keyboard that activates it
- Distance – How far it moves back while holding the button
- Return Speed – How fast it moves back while holding the button
- Launch Speed – How fast it moves pushes forward when you release the button

BOARD LIGHT



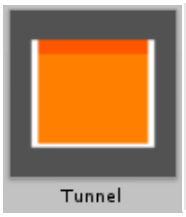
Lights up when the ball runs over it, or at specified intervals.

Board Light component:



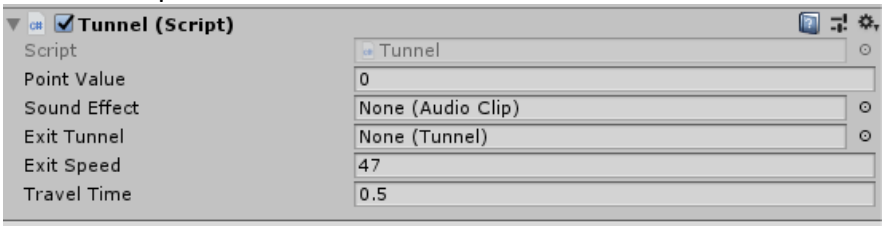
- Point Value – Number of points awarded when a ball collides with it
- Sound Effect – The sound played when a ball collides with it. Click the circle to the right of this to view all sounds.
- Light Mode
    - Proximity: Lights up only when a ball runs over it
    - Blink: Continuously blinks regardless of whether a ball runs over it
- On Time: Time in seconds of how long the square will light up.
- Delay Time: Time in seconds of how long it will wait to light up once triggered (either by a ball in Proximity mode OR when starting the game in Blink mode)

TUNNEL



Holds the ball for a short some time and then shoots it out again. It can also teleport the ball to another specified tunnel.

Tunnel component:



- Point Value – Number of points awarded when a ball collides with it
- Sound Effect – The sound played when a ball collides with it. Click the circle to the right of this to view all sounds.
- Exit Tunnel – The tunnel the ball will be teleported to when it enters. Leave it as *None (Tunnel)* to make It exit the same tunnel it came in.
- Exit Speed – How fast the ball will be launched out.
- Travel Time – How much time in seconds that the ball will stay inside.

EDITING PREFABS

If you have multiple objects created from the same prefab, you might find it a little difficult to change all of them at once. To solve this, we can edit the prefab they copy from directly.

Let's say I want to change all my bumpers to be a certain colour. Instead of changing each individual bumper, I can edit the prefab they copy from. All instances will use this new colour by default.

- Navigate to the prefabs folder
- Double click on the prefab you want to edit. Scene view will now look a little different
- In the Inspector, change a parameter you wish to set to a new default. I am changing the default colour of my bumpers.
- When you're done, click the left arrow near the top of the Hierarchy window.

SCRIPTS

You can add additional scripts as components to objects or prefabs. This will modify the behaviour in some way.

To add a script to an object:

- Navigate to the Scripts folder in the Project view.
- Click and drag one of 3 scripts from the Project window over an object listed in the Hierarchy window.

There are 3 scripts you can add to any object:

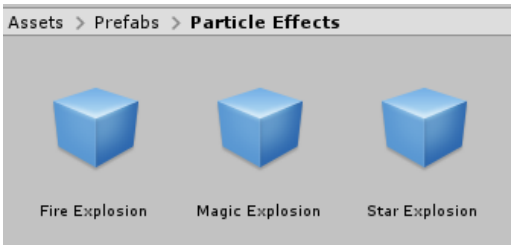
ColourChanger
Changes the object to a random colour when the ball collides with it.

Rotator
Constantly spins the object.

- Speed - How fast the object should rotate in clockwise motion.

ParticleEmitter
Releases some visual effects when the ball collides with the object.

- Particle effect - What effect you want to appear.
    - In the Project view, navigate to the Prefabs>Particle effects folder to view the list of particle effects you can use.



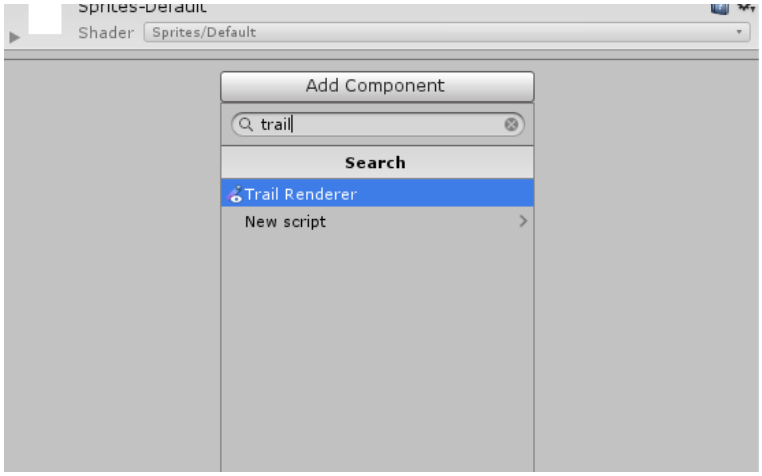
- Click the object with the ParticleEmitter component so that you can view its Inspector.
- Click and drag the one you want to in the slot of the Particle Effect of the ParticleEmitter component. It is set to None (Particle System) by default.

TRAIL RENDERER

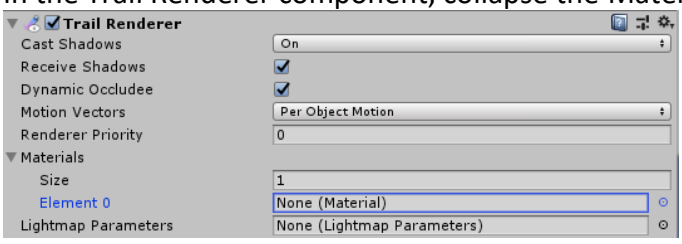
It's possible to add a trail behind the ball as it moves as a neat visual effect:

Here's how to set it up:

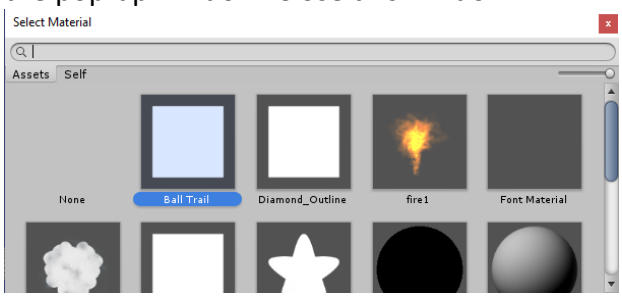
- Navigate to the prefab folder and double click the ball. This will change your scene view.
- In the Inspector window click **Add Component**
- Search for "Trail Renderer" and click it



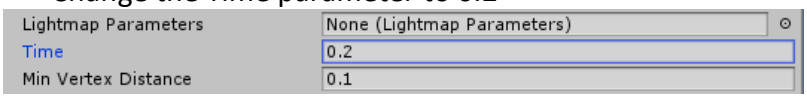
- In the Trail Renderer component, collapse the Materials list



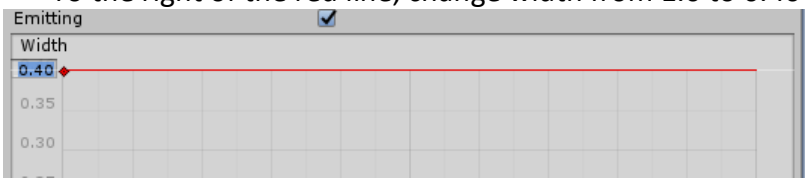
- Click the circle to the right side of the Element 0 parameter and select Ball Trail from the pop-up window. Close this window.



- Change the Time parameter to 0.2



- To the right of the red line, change width from 1.0 to 0.40



VARIATION

Using all the features above, here are a few examples of what you could create!