unity

Unity Lander 1

Instructor Guide

An instructor's guide to
getting started with Unity

Table of Contents

Project Introduction

Overview

Get started creating with Unity, professional game development software. The lesson in this guide is a great first step toward learning all of what Unity can do--from making games to making 3D models and environments, VR and AR. In this lesson, your students will play a Moon Lander game and create their own unique level for the game. (**Figure 1**).

Figure1: Unity Lander Moon Lander

Introduction

Create a level for a physics-based game built with professional game making software. Download and import the Mouse Moon Lander package that gives you all the assets you need to build your first Unity game while learning how to use the Unity.

Educators - It is highly recommended to read through each project steps as you prepare for facilitating this project.

Intro to Unity

Background

Overview

Learners become familiar with Unity game making software by rearranging obstacles to create their own unique level for an existing game. This projects takes approximately 60-70 minutes of instructional time.

After completing this activity learners will have everything they need to do the Unity Lander series of activities. Complete both activities and learners will create a Moon Lander game where they will move a lander vehicle through treacherous terrain to land in a specific location.

Course

This project is part of Mouse's Serious Games course. If learners finish this project successfully, they will have completed 1 of 2 Unity projects from Mouse's Serious Games course. The full Serious Games course is a game design course focusing on creating games that try to do more than just entertain using Scratch, Inklewriter or Unity.

Mouse is a youth development non-profit with the that teaches technology with purpose. To learn more about other Mouse courses, please visit our Course Directory.

Associated Projects

The other project associated with this one is:

- Unity Lander 2

Outcomes

You will be able to:

- Create and activate a Unity account
- Install Unity and components
- Create and save a Unity Project
- Save and open a scene in Unity
- Import custom assets assets into Unity
- Reset the Unity Window Layout
- Enter and exit Play Mode
- Move GameObjects around the Scene view
- Activate and Deactivate GameObjects
- Save a Unity Scene with a new name.
- Move the Scene View around a Project
- Create and modify GameObjects in Unity
- Modify a GameObject's Transform component in Unity

- Add Collider components to GameObjects
- Build a player character that you can move with a keyboard
- Create nested Parent/Child GameObjects in the Unity Hierarchy
- Add scripts to Unity GameObjects
- Adjust physics for GameObjects
- Create a Prefab

Standards

All Serious Games activities have been aligned to Common Core, Next Generation Science (NGSS) and International Society for Technology in Education (ISTE) learning standards. You can find all of those alignments here:

- Common Core & ISTE
- NGSS

Gear

- Unity software
- Laptop or desktop computer that meets the following system requirements
- A Unity for Education license obtained by Unity prior to installing the software (if you are doing this in a classroom setting) OR a Unity Personal Edition license (if you are doing this activity on your own)
- The Mouse Lander Package available here: moon-lander-package
- The Unity Manual

Activity Flow

- **Step 1: Intro (1 minute):** Introduction to the project. See background for information about set-up before starting.
- **Step 2: Create a New 2D Game (3 minutes)**: Create a new 2D game.
- **Step 3: Download the Moon Lander Asset Package (10-15 minutes):** Download and import the custom asset package for Mouse Unity activities.
- **Step 4: Layout (1 minute):** Set our Default layout setting for Unity.
- **Step 5: Loading and Playing a Scene (5 minutes):** Load a scene file to play a Unity game and discover a broken level.
- **Step 6: Play a Scene (5 minutes):** Play a final version of the game that you will create in Unity.
- **Step 7: Camera and Background (3 minutes):** Set a background color for your level.
- **Step 8: Activating and Deactivating GameObjects (2 minutes):** Temporarily deactivate the main camera.
- **Step 9: Adding GameObjects to the Hierarchy (3 minutes)**: Add a Ship GameObject to the

Hierarchy.

- **Step 10: Moving Around the Scene View (2 minutes):** Move the camera around the Scene View to zoom in and out on GameObjects as you edit them.
- **Step 11: Transforming GameObjects (3 minutes):** Reposition and scale objects.
- **Step 12: Colliders (3 minutes):** Set colliders that detect when this GameObject touches another GameObject.
- **Step 13: Parents and Children (3 minutes):** Create a parent/child relationship between two GameObjects.
- **Step 14: Landing Legs (3 minutes):** Create landing legs for our lunar lander that have
- separate Colliders but have the same parent as our ship.
- **Step 15: Moving the Player (3 minutes):** Add scripts to our player that allow us to control it with a keyboard.
- **Step 16: Gravity, Force and Drag (5 minutes):** Adjust the physics settings to our liking for our game.
- **Step 17: Player Prefab (1 min):** Make a Prefab of our ship so that we can use it in another game
- **Step 18: Level Design (5-10 minutes):** Create a new level by remixing a broken level.
- **Step 19: Save the Scene (1 minute):** Save the changes to the Unity game.
- **Step 20: Beta Feedback** (optional)

Educators: These times are rough estimates. Be sure to look to the content of the project's steps to help you plan the timing, flow and content of the lesson for your specific group of learners.

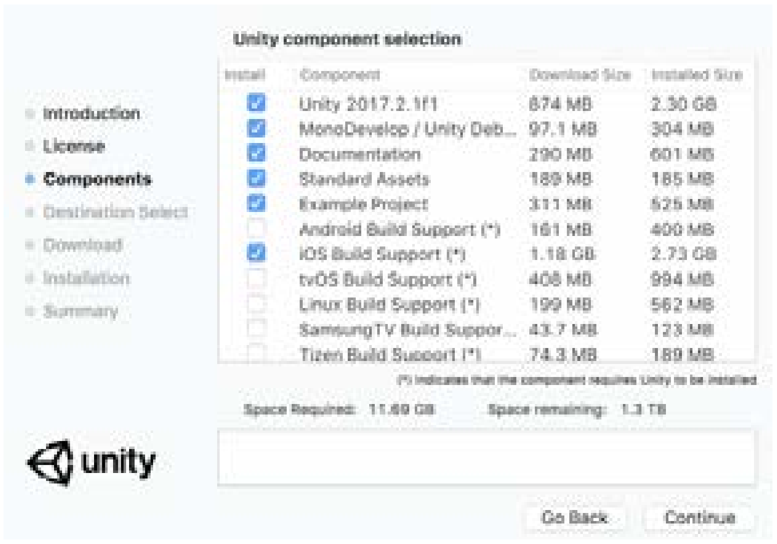
If you need to break the project up over multiple sessions, we suggest the following:

- Session 1: Steps 1 - 11, then skip to step 18 to save what they've made.
- Session 2: Steps 12 - 19.

Prep

- Apply for a Unity Education license for classroom use of Unity. It may take a day or two to get a response from Unity so please do this with plenty of time before facilitating this project in a classroom.
- Create a Unity ID
    - Use a generic email address for this ID, one that you are comfortable letting all students log into.
    - If you want to use Unity for personal use you can get the free Unity personal edition.
- Once you get your Education license, will have a free access to Unity Pro, so go ahead and download the Pro version of Unity. Ignore any screens asking for billing information.
- While installing Unity you will be asked to select which components you would like to select (**Figure 2**). This will affect what kind of devices you can build your finished projects

for.


Unity component selection

Install	Component	Download Size	Installed Size
✓	Unity 2017.2.1f1	874 MB	2.30 GB
✓	MonoDevelop / Unity Deb...	97.1 MB	304 MB
✓	Documentation	290 MB	601 MB
✓	Standard Assets	189 MB	185 MB
✓	Example Project	311 MB	525 MB
	Android Build Support (*)	161 MB	400 MB
✓	iOS Build Support (*)	1.18 GB	2.73 GB
	tvOS Build Support (*)	408 MB	994 MB
	Linux Build Support (*)	199 MB	562 MB
	SamsungTV Build Suppor...	43.7 MB	123 MB
	Tizen Build Support (*)	74.3 MB	189 MB

(*) indicates that the component requires Unity to be installed

Space Required: 11.69 GB   Space remaining: 1.3 TB

Figure2: Unity Lander Moon Lander

Make sure you have either WebGL selected if you want to make projects that are playable in a WebBrowser, and Mac or PC selected depending on which Operating Systems you would like to be able to play your games.

- Create a Unity ID for your class using a generic email, like your school email address or a class email address. If you or your students want to use Unity on your own you will need a different email address for your personal edition.
- If you are using Unity at home download the personal edition, which is free as well. You will need to create a Unity account and for that you will need an email address.
- ***Downloading and installing Unity may take 30-60 minutes depending on your internet speed and computer speed***

Glossary

- **Asset:** A file that is part of your game. For example, an image file, a code file, a sprite sheet, etc.
- **C# (pronounced C sharp):** A programming language that uses classes. Many Unity scripts are written in C#.

- **Focus:** When using Unity, bringing something into focus means it appears in the inspector.
- **GameObject:** A GameObject in Unity can be an object in your game, a part of the background, or it can be visually empty but contains information, for example text from your user interface, a menu, or a score counter.

- **Game view:** A window that shows what your game looks like to a player.
- **Hierarchy:** The active GameObjects in your scene.
- **Inspector:** Shows you details about an asset or GameObject. Allows you to set layers, add tags, and add components.
- **Prefab:** A GameObject template that allows you to create many copies of GameObjects with the same specifications.
- **Project:** Your whole game is your project. Your project can be made of one or many scenes.
- **Project window:** The window that shows you the folders, scenes and assets that are in your game.
- **Scene:** A scene is a section of your game. It can be a single sequence, a level or some other slice of your game. A game can have only one scene, or it may link several scenes together.
- **Scene View:** A window in Unity that allows you to build the world of your game, where you can select, position and manipulate GameObjects. It shows you all the active objects in your Hierarchy in a visual representation.
- **Script:** A piece of code that can be attached to a GameObject in Unity.
- **Sprite:** A 2D graphic.

Evidence

What will learners do at the end that will demonstrate what they've learned?

At the end of the project, learners will have used Unity to create a basic digital game scene in which a spaceship must navigate through asteroids to reach a platform on the other side of the screen.

Evidence Checklist

What should teachers look for to know that the evidence above is a success? You may realize after writing these that you need to change/edit the learning outcomes at the top.

Learners have met target outcomes if:

- ☐ Learners have changed the background color of the remix-me scene.
- ☐ Learners have tested their new version of the remix-me scene multiple times to make sure the level is not too challenging but not too easy.
- ☐ There is a new scene saved in the Scenes folder that is winnable but challenging.
- ☐ The Hierarchy of the new scene contains the Player GameObject, which has three children: LeftLander, RightLander and Ship in addition to other GameObjects.
- ☐ Each child of Player has a separate 2D Collider.
- ☐ The top level Player GameObject has two scripts attached as components: PushWithButton and RotateWithArrows.

- ☐ Gravity is set to a number other than 1.
- ☐ Movement settings are adjusted to taste.

Debrief Questions

- What is a Scene in Unity?
- What is an Asset?
- What does the camera do?
- What color did you choose for your background?
- How difficult did you make your game?
- What do physics have to do with your game?
- Why would a ship keep spinning forever in space?
- How did you like your controls? How did you customize them?
- What is a script?
- What is a Collider?
- What if you replaced the spaceship with a different Ship, what would it be?
- If you replaced the asteroids with something, what you would replace them with?

Extension ideas

What can learners do if they finish the project early? Are there ways educators can level the activity up or down?

- The unity-moon-lander-package is a selection of assets taken from the Unity Playground project. The full playground project contains many more sprites and and scripts for learners to make games. The collection of scripts are useful for allowing learners to play and experiment with common behaviors like keyboard movements and colliders as triggers that can be used to create a variety of different games.
- For those who want to try a tutorial featuring the newest features of the Playground Project try this tutorial on new features.
- Additional Unity tutorials can be found here

Project Steps

Introduction

Instructor note: *Repeat the Introduction and the first step.*

Create a level for a game built with professional game making software, Unity! (Figure 3) Download and import the Mouse Moon Lander package that gives you all the assets you need to build your first Unity Game. Make sure you check the Project Background for information on downloading Unity and getting a Unity ID.

Figure 3: Unity logo

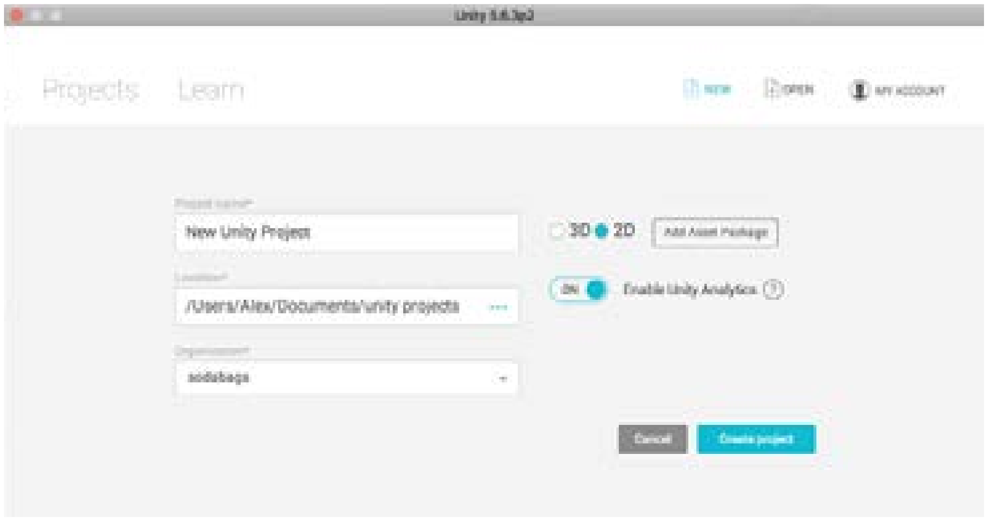
Create a New 2D Game

Unity is an all-in-one editor used to make games, animations and virtual environments.

Many games are built with Unity because it is available for both Windows and Mac and can build games that run on almost any platform imaginable. Plus, for students and hobbyists, Unity is free.

For this project and any other Mouse Create projects that use Unity you will need to download and install Unity. See the Project Background for help on how to do this.

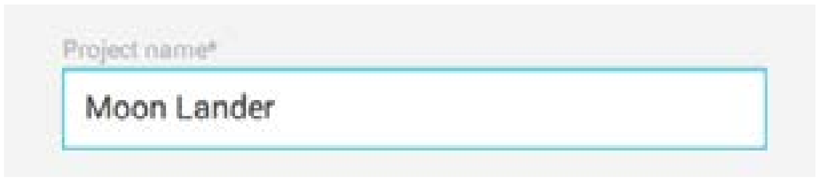
Once you've logged into Unity, create a new game by clicking the "New" button.



Project name*: New Unity Project
Location*: /Users/Alex/Documents/unity projects
Organization*: sodabags
○ 3D ● 2D
ON Enable Unity Analytics

Figure 4: Unity New Project Screen

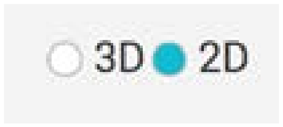
Title your game Moon Lander or make up your own special name.



Project name*: Moon Lander

Figure 5: Project Name Field

When the screen appears make sure you have "2D" selected for the game's type.



○ 3D ● 2D

Figure 6: 2D / 3D toggle, with 2D selected

Make sure you know where you are saving your game, you'll be creating a folder that contains all your game information and you might need to access that folder later.

If this is your first time using Unity, it is best to create a folder for all of your Unity projects called "Unity Projects".

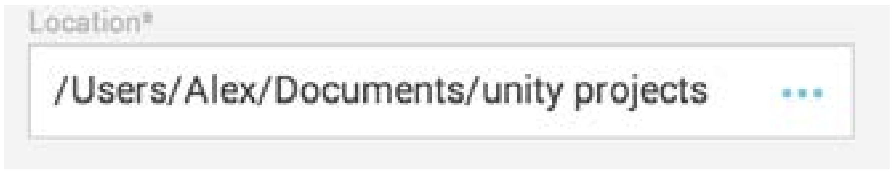


Figure 7: Mac Example

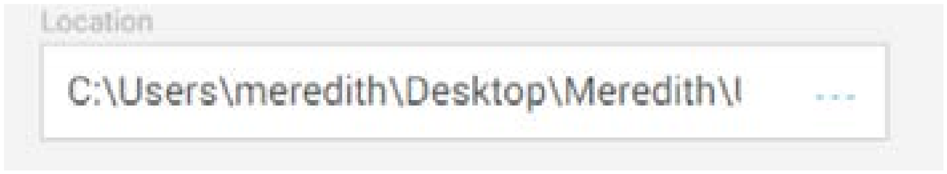


Figure 8: PC Example

Facilitator Note:

Unity automatically creates a new folder to contain each project. Inside that folder will be a number of other folders that hold that projects sprites, scripts, Prefabs, scenes and other assets. If multiple learners are sharing the same device make sure they each create separate projects and not scenes, in order to keep their projects separate. Every time they create a new project they should see the New Project scene, and make sure when they are opening a saved project to always select "Open Project" from the File menu.

Click Create Project to continue.

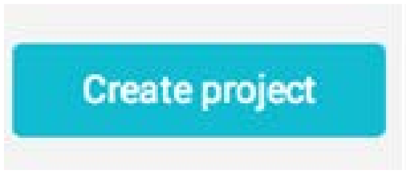


Figure 9: Create Project button

Facilitator Note:

The organization field denotes shows which organization the active Unity account is connected to. This will most likely be your school.

Download the Moon Lander Asset Package

Important Note: The asset package in this step was made in Unity 2017.3 and may have errors if opened in an earlier version of Unity. Please update your version of Unity to 2017.3 or later to complete this project.

In this this project, you're going to learn some Unity basics by remixing a Moon Lander game Mouse has already created. To do this, you will need to download the Mouse "Moon Lander" package of assets from here: [moon-lander-package]

Assets are the files that make up your game. In Unity you will take the assets we have provided you with and turn some of them into GameObjects, the basic building block of Unity. Other Assets are scripts that you will attach to those GameObjects to give them behaviors.

Once the package has downloaded, you should a new file that ends with ".unitypackage." Make sure this is saved to a place you can find on your computer.

In Unity, go to the Assets menu at the top of the screen and select Import Package, then select Custom Package. Find the "mouse-moon-lander-kit.unitypackage" file and select open. You will then get a menu screen of all the files in the folder.

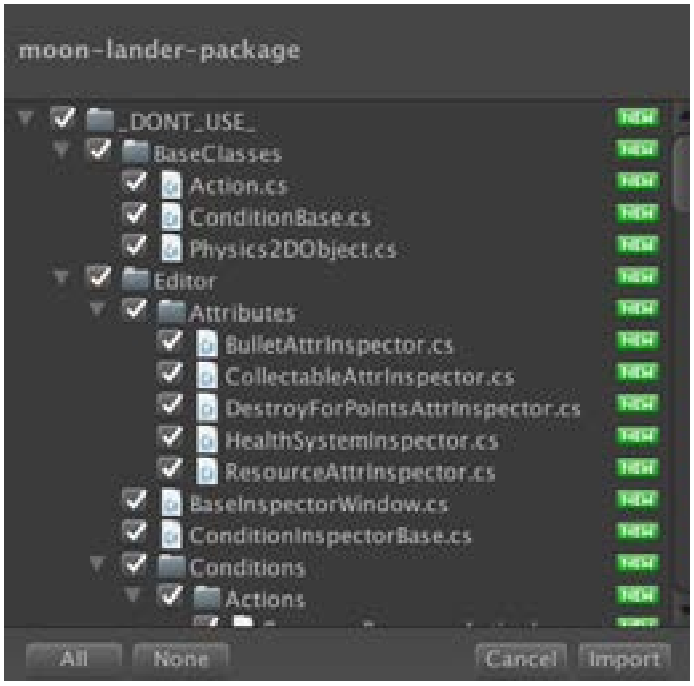


Figure 10: Import Package screen after the Mouse custom package is selected

Click All to make sure you have everything selected. Click "Import" to import them all. Unity will import all the assets you need to make your game.

Facilitator Note:

This shows how to import a custom asset package into Unity, but many types of other assets can be imported to Unity. You can find more information about importing assets in the manual.

Congratulations, you have imported everything you will need to create your Moon Lander Game.

Facilitator Note:

The Mouse “Moon Lander” asset package is a selection of assets taken from the Unity Playground Project, a series of assets which includes pre-written scripts that allow you to create working Unity games without having to learn to code in C#. The full Playground Project has a lot more sprites and other assets than the Moon Lander project. If you would like to experiment with it then go to the Playground Project’s GitHub page click “Clone or Download”, then click “Download ZIP” to download the entire folder.

Layout

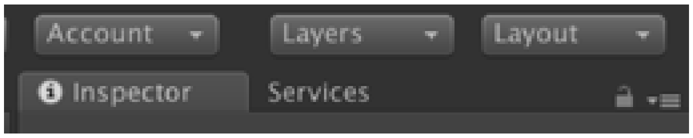


Figure 11: Layout dropdown

Let’s start by making sure we’re working in the right layout. Unity has a number of different windows that allow you to edit different parts of your game. We are going to be using the Default window layout to make our game, if your screen layout looks different from ours just click on the “Layout” dropdown in the top right corner and select Default to select the Default layout. If you are in a different layout you will see the name of that layout where it says “Layout.”

Depending on whether you are using the Personal version or Education version you may notice that Unity has a different background color. For what we are going to do in this project, it doesn’t matter which you have. Don’t worry if the screenshots don’t look exactly like your version of Unity.

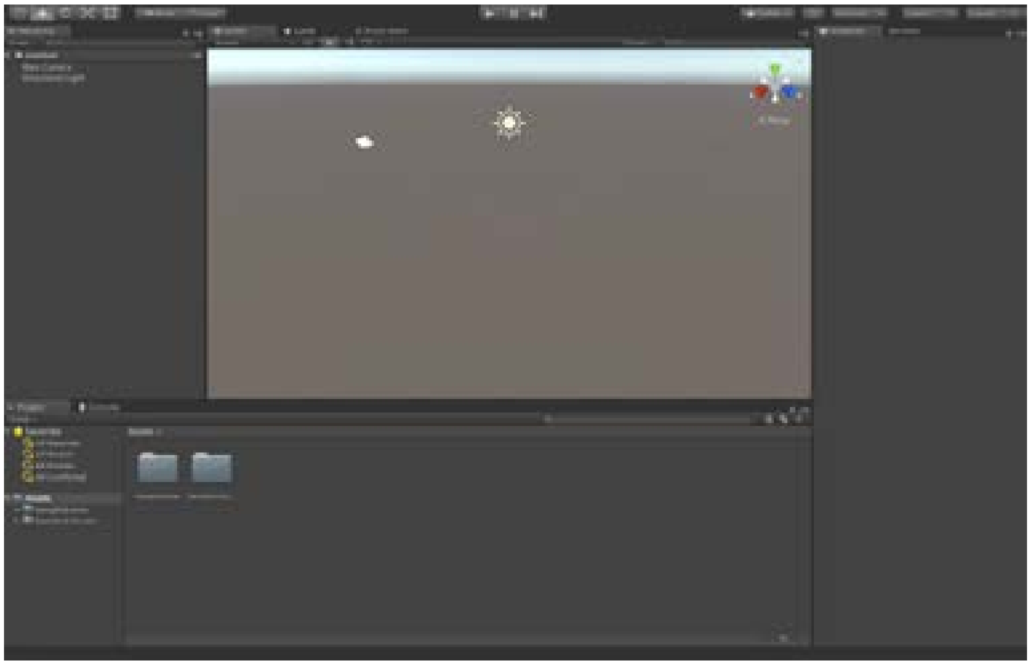


Figure 12: Unity Pro/Educational Version

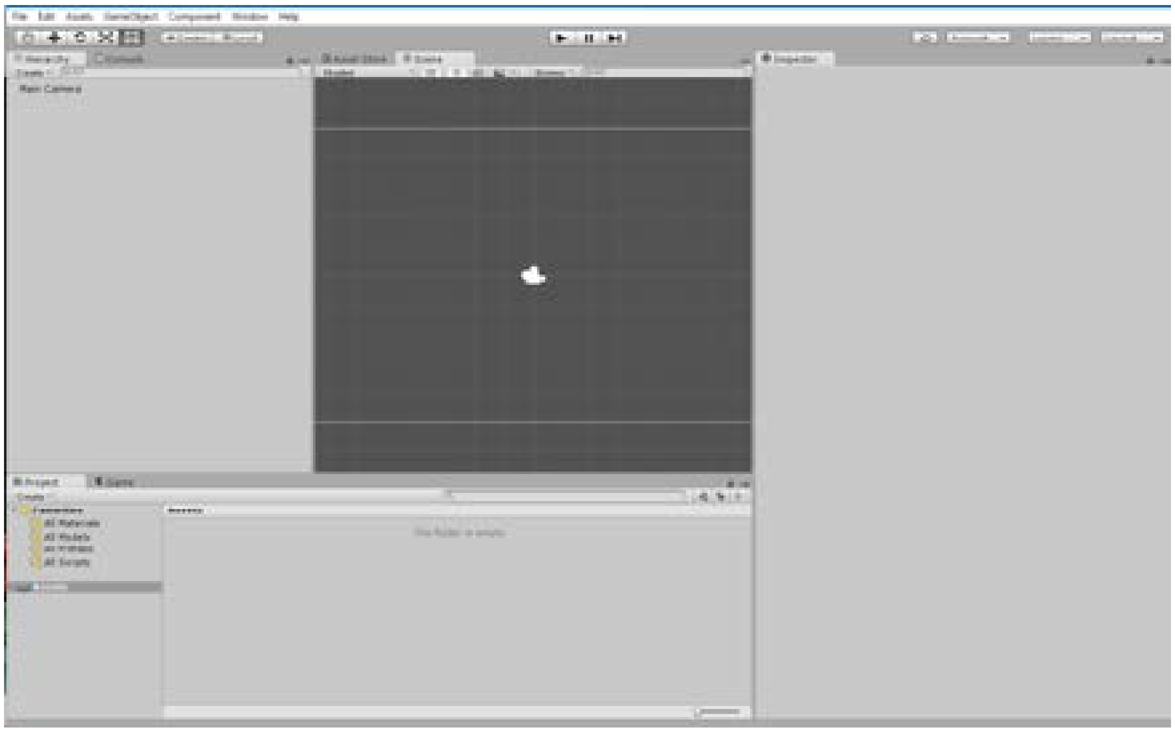


Figure 13: Unity Personal Edition

Loading a Scene

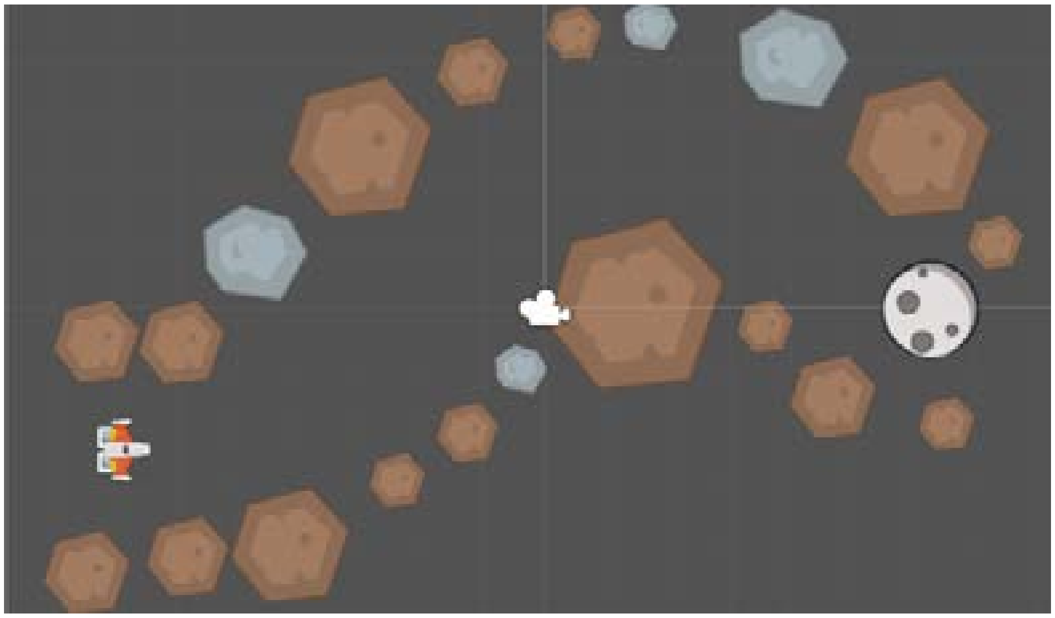


Figure 14: Level1 scene

Now we'll take a closer look at **Scenes**. Scenes contain the environments and menus for your game. Think of a scene as a single level of your game. Each scene can hold different player characters, enemies, background, menu options, and more. Once you beat a level in a Unity game, you might want a new scene to load that has the next level inside it.

Look at the bottom of your screen. This is the **Project window**. It shows you the Assets in your project, which are the various files that will make up your game. You should see a number of new folders in there.

In the Assets folder is a folder called Scenes. Go inside Scenes by double-clicking on it and find a file called **level1**. Double click on the scene to load it.

Facilitator Note:

Scenes files have the file extension .unity when viewed in the Windows File Explorer or Mac OS Finder. While looking in the Unity Project Window, if you select a file you can see it's full name and file extension in the very bottom of the Project Window.

In the upper left corner of the screen is the **Hierarchy window.** This window shows all the active objects in your game.

Click on the starting platform right below the spaceship to select it. That will highlight the platform in the Hierarchy. You should see that "Start" is highlighted in the Hierarchy.

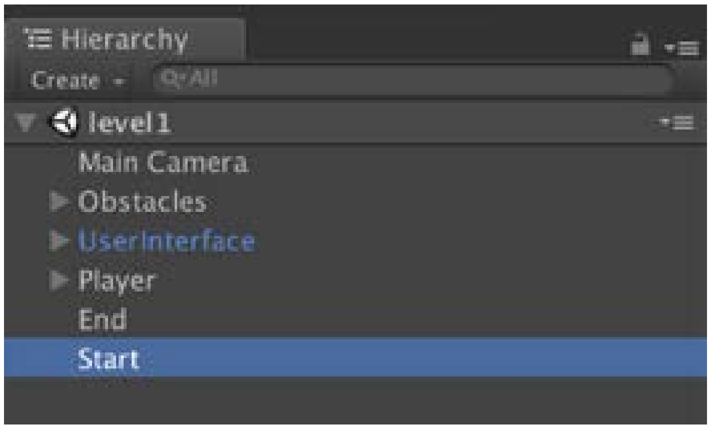
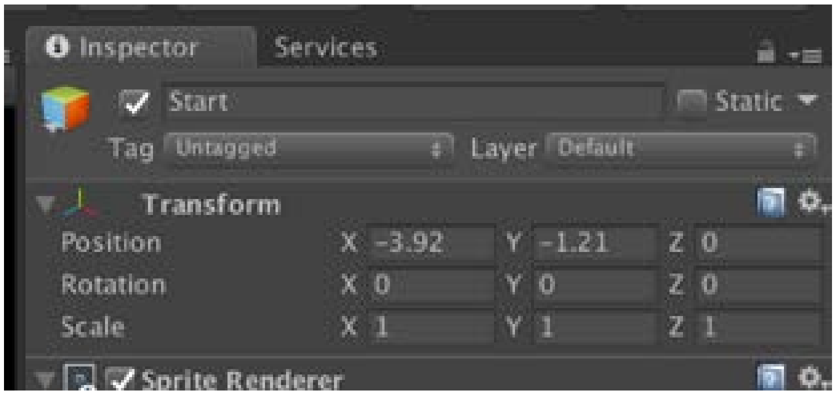


Figure 15: Start selected in Hierarchy

Selecting the Start platform also makes it appear in the **Inspector window** on the far right hand side. You may need to select the Inspector tab if you don't see it. Once a GameObject is in focus you can use the Inspector customize it or add **Components** which controls how GameObjects look and behave in your game.



Start
Tag Untagged   Layer Default

Position	X -3.92	Y -1.21	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Figure 16: Start in focus

Whatever is viewable in the Inspector is called "**in focus**." It's important to keep in mind what is currently in focus, and how bring that back into focus. It can be easy to accidentally lose your focus on something while opening folders or going back and forth between GameObjects in the Unity Editor.

Play a Scene

Let's try playing the game in the scene.

This is a simple game where you start with a Moon Lander sitting on one platform and your goal is to safely move your ship to the other platform on the screen. Once you touch the second platform you'll get a message letting you know you beat the game.

To play the game, enter Play mode by clicking the triangle icon in the center of the top of the screen (it looks like a Play button).

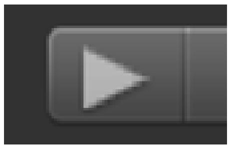


Figure 17: Play Mode button

Press the spacebar to use the ship's engine to push the ship forward. Use the left and right arrow keys on your keyboard to steer the ship. You don't need to press the spacebar for long, just tapping it lightly will move your ship forward.

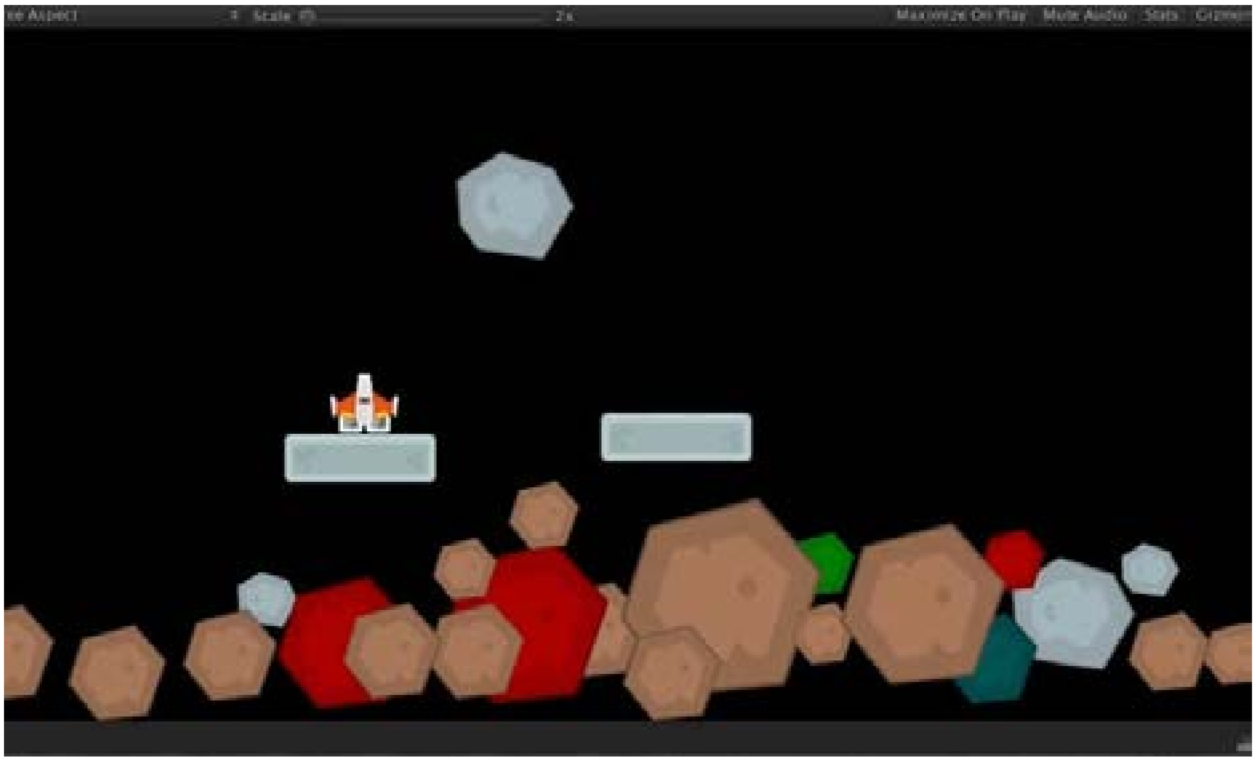


Figure 18: Ship completing Level 1

Exit Play Mode by clicking on the play button again. If Unity asks you to save the scene you can select “No”. It’s important to fully exit play mode because any changes you make to your project will not be saved while you are in Play Mode.

Now that you’ve played this game, you’re going to remix the scene to make your own version of it.

Double click on the Unity Scene called **remix-me** to open an unplayable version of the game.

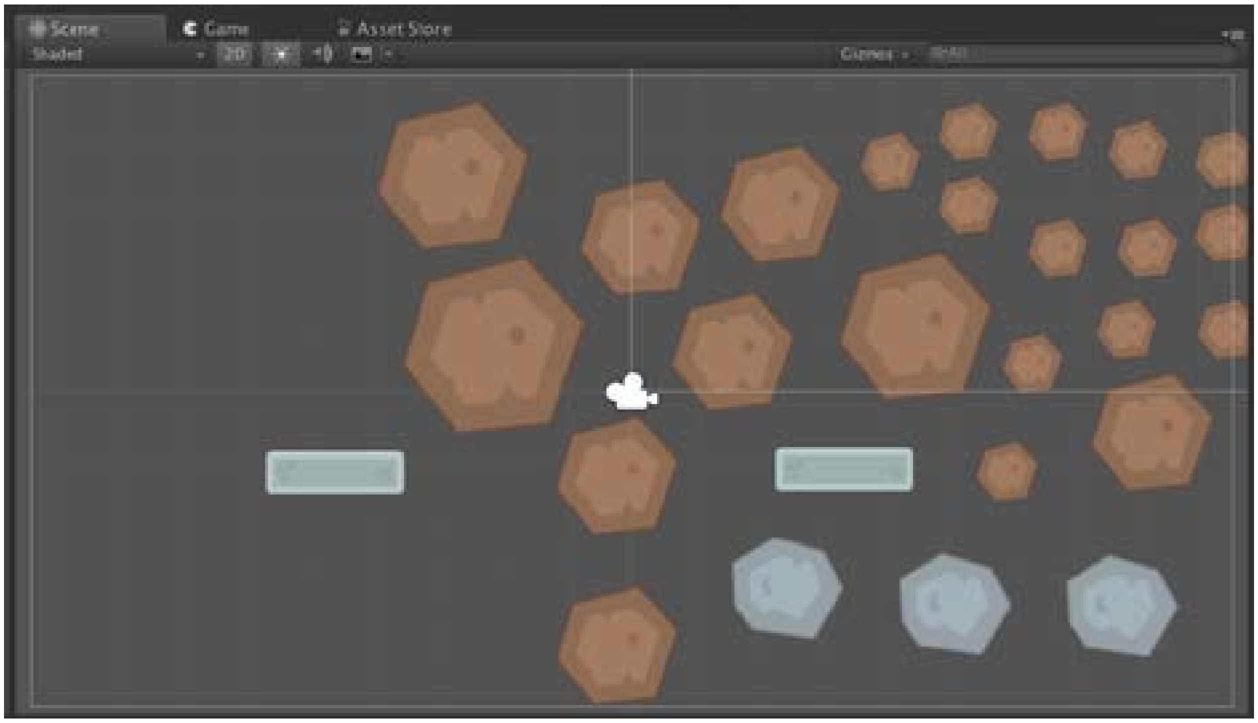


Figure 19: Screenshot of the broken level

This game is impossible to play! There’s no player character and even if there was it couldn’t get to the other platform. In the next few steps you’ll learn how to use Unity tools to fix it and create your own version of Moon Lander.

Camera and Background

The first thing you're going to want to know about to fix the game is the Main Camera. Every Unity scene starts by default with a Main Camera, which controls what your players will see when they play the game. In this step we're going to change the background color that our camera sees.

Select the Main Camera GameObject in your Hierarchy window on the left. (Remember, the Hierarchy Window shows you what is active in the current scene of your project.)

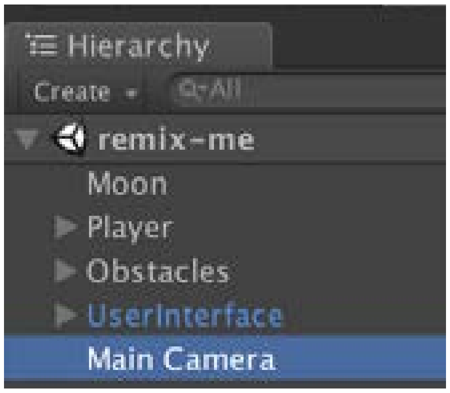


Figure 20: Main Camera GameObject selected in the Hierarchy

In the Inspector for the camera (on the right side) is an option called Background. You can use this to control the color of your background.

Some games use images for their backgrounds but for our game we're going to use a solid color. If you look at the Inspector you will see the background is currently set to white.

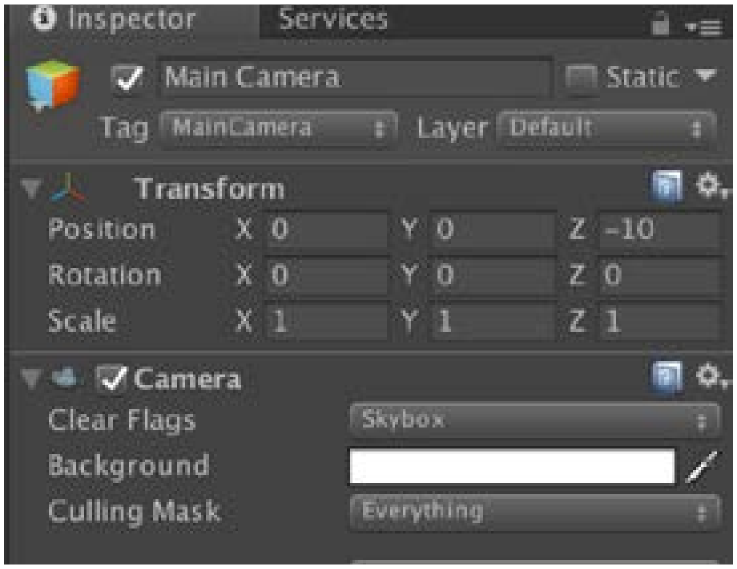


Figure 21: Inspector with the Main Camera in focus

Enter Play Mode for **remix-me** and see what this solid white background looks like in our game.

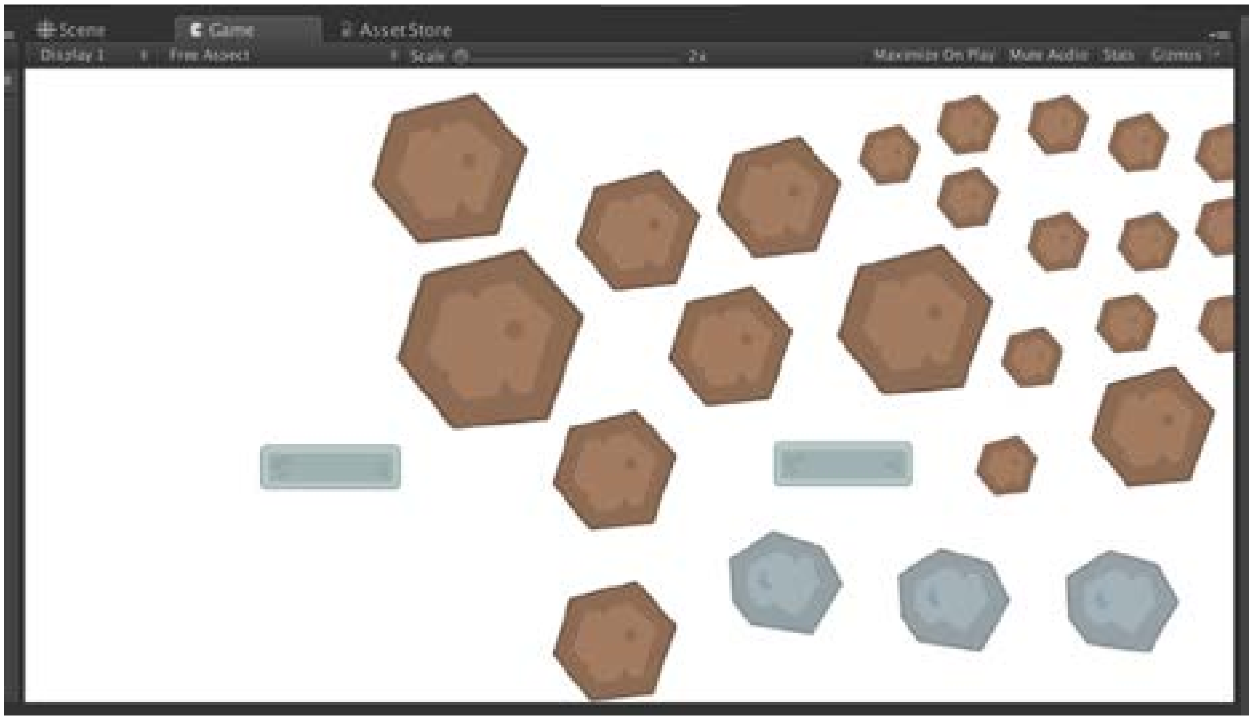
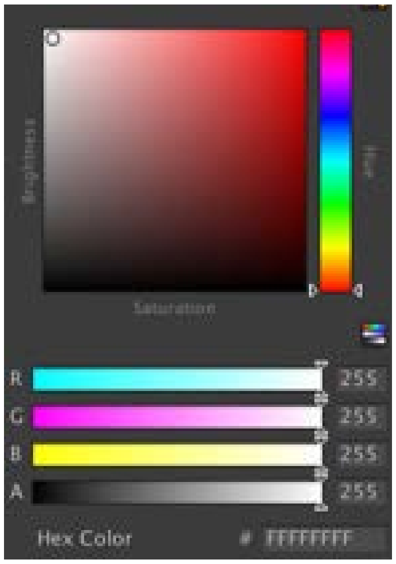


Figure 22: Remix Me Scene View

It doesn't look much like space. Exit Play Mode and click on the Background area in the Inspector to open up the color picker and select a new background color.



R 255
G 255
B 255
A 255
Hex Color # FFFFFFFF

Figure 23: Color Picker

Activating and Deactivating GameObjects

Notice how there's a little white camera in the middle of the screen? Let's make it disappear for a while so it's not in the way while we are editing our level.

Make sure you are in Scene view and not in Game view during this step (and whenever you have the camera turned off). Scene view allows you to look behind the scenes of your game and edit your GameObjects. Game view shows you the perspective from your Main Camera so it won't work while you have the camera deselected

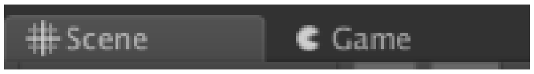


Figure 24: Scene View and Game View tabs

Make sure the Main Camera is still in focus. Now that we've set our background, we're happy with our camera settings. So we're going to deactivate this GameObject for a little while.

Look in the Inspector window. Do you see the check mark next to the title of the Main Camera GameObject?

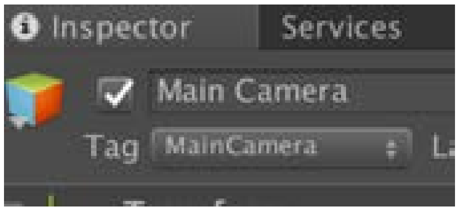


Figure 25: Inspector with the Main Camera in focus with an active check mark

For now, uncheck that box to deactivate the Main camera. When an object is not active it is invisible, but none of its information is gone. We will need to reactivate our camera when it's time to test our game. You can do this with any GameObject that in the scene, just remember to reactivate them when it's time to test your game.

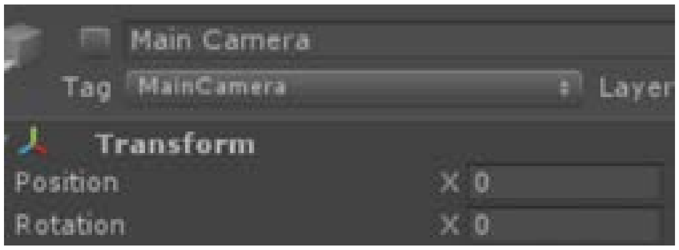


Figure 26: Main Camera Unchecked and Deactivated

For the next few steps we recommend you temporarily deactivate the **UserInterface** GameObject as well.

Adding GameObjects to the Hierarchy

First let's create a player character. We're going to create a ship that you can control with your keyboard. It will be made up of several GameObjects.

GameObjects are the building blocks of Unity. If you have experience with Scratch, a GameObject is similar in concept to a Scratch Sprite. In Unity the term "Sprite" refers to just a 2D graphic file, while a GameObject can be a player character, a part of the background, or an empty placeholder that keeps track of something, like a the score or a piece of code that triggers an explosion, or just a small part of one of those things.

Make sure you have deactivated the Main Camera in the last step.

In the Project window double click on the Sprites folder and then double-click on the **Ships** folder.

Drag the ship into the Hierarchy window on the left side of the screen. This will make it an active object in our scene.

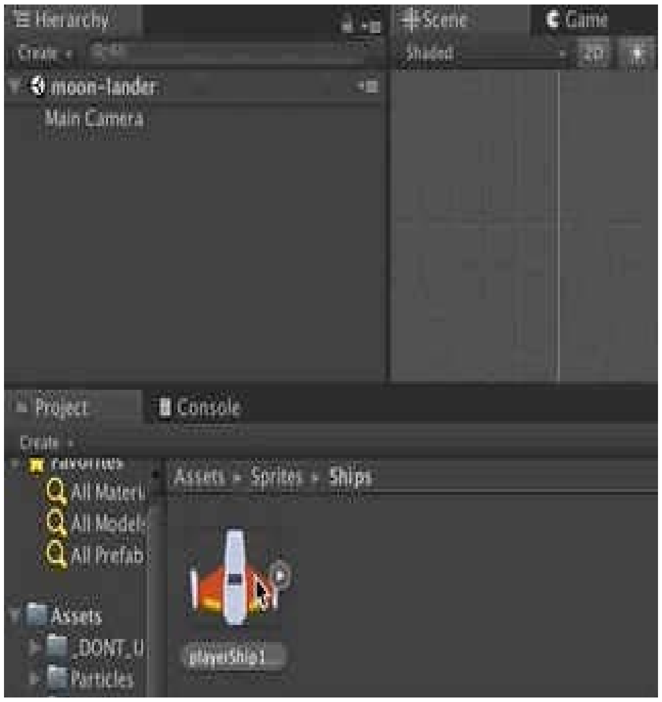


Figure 27: Clicking and dragging the PlayShip sprite from the Project Window to the Hierarchy

Now select the ship's name in the Hierarchy.

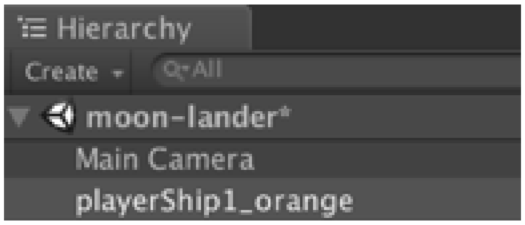


Figure 28: playerShip1_orange selected in the Hierarchy

Then right-click (or control+click on a Mac laptop) and select “rename.” Give it the new name “Ship.”

Facilitator Note:

On a Mac you can use control click instead of a right click or simply press the return key while you have GameObject highlighted that you would like to rename. Hit the return key again once you are done naming it.

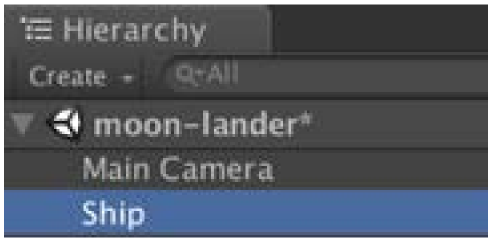


Figure 29: Ship selected in the Hierarchy

Congrats we’ve created our first GameObject!

Facilitator Note:

If learners want to customize their ship they can adjust the color of the ship in its Sprite Renderer component in the same way that we changed the Main Camera’s background color. Advanced learners can use their own graphic for a ship. All the most common image file types are supported by Unity (BMP, TIF, TGA, JPG, and PSD). Learners can simply drag and drop an image into the Project window or they can add it by going to the Asset window at the top of the screen and selecting Import New Asset.

Moving Around the Scene view

A lot of the time we spend in Unity will be using the Scene view. The Scene view lets you see and change the world you are creating. You will use the Scene view to select and position pretty much everything that is in your game. When you enter Play Mode, it brings up the Game view which shows you what the game will look to your player and lets you test your game. You can switch between the Scene view and the Game view with these tabs in near the top of your screen, near the center (although if you turned your camera off, the Game view won't work).

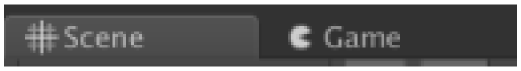


Figure 30: Scene View and Game View tabs

There are a few ways to look around in the Scene view:

Use the Hand tool:

When the Hand tool is selected (shortcut: Q), it will look like this:

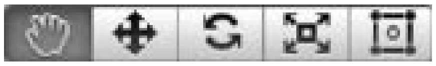


Figure 31: Hand tool shortcut selected

Once the Hand tool is selected you can click and drag to move the what you can see.
When the Hand tool is selected you can also zoom in and out on a PC by holding down the Alt key, right-clicking and dragging to zoom the Scene view. On Mac you can hold down Control and click and drag instead. While you are zooming your toolbar on the upper left will look like this:

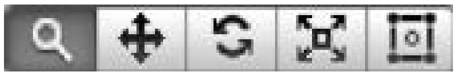


Figure 32: Zoom tool hotkey selected

Zooming:

You can zoom in on an object by selecting the object in the scene view, and, while your mouse hovers over the Scene view window, hitting the F key. Unity will zoom in and frame whatever you have selected so that it takes up most of the screen. This can be useful when you want to zoom in to edit specific details. To zoom back out. you can select the Main Camera and hit the F key while the cursor is in the Scene view and Unity will frame the whole Main Camera's field of view in the Scene view.

Select the ship sprite we just added to the Hierarchy and hit the F key while the mouse is over the Scene View to put it into the center of our screen and zoom in on it.

Hit the F key while the cursor is hovering over the Scene view to zoom out again and see your Main Camera's entire field of view.

Transforming GameObjects

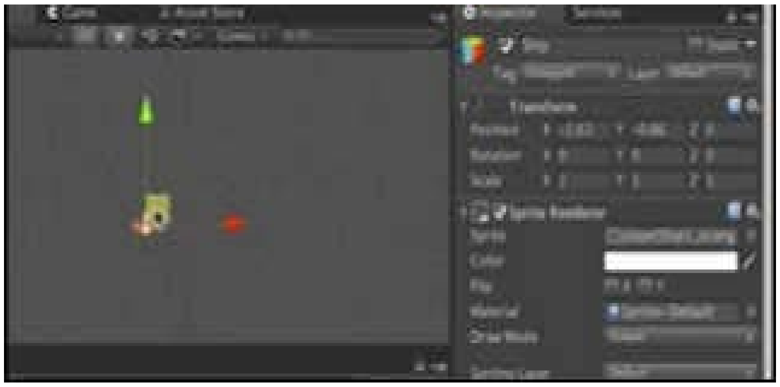


Figure 33: Ship GameObject with the Move Tool and the Transform Component

GameObjects can be controlled by using different components. In this step we will use the **Transform component** to change the size and position of our GameObjects. Every GameObject (even empty ones) start with a Transform component.

Select the Move tool from the upper left corner of the screen and try moving your ship around the screen.

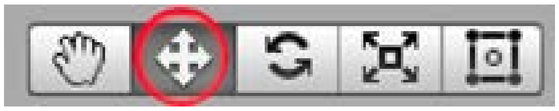
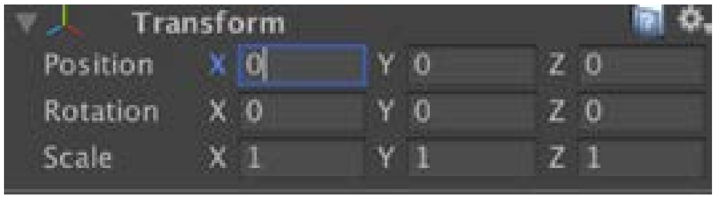


Figure 34: Hotkeys with a red circle around the Move Tool

Click and drag on the red and green arrows that appear behind the object to move it in a single direction, or drag the blue square to move it freehand around the scene.

Now look at the Transform component in the Inspector window on the right side of the screen. It should be the first component you see.



Transform

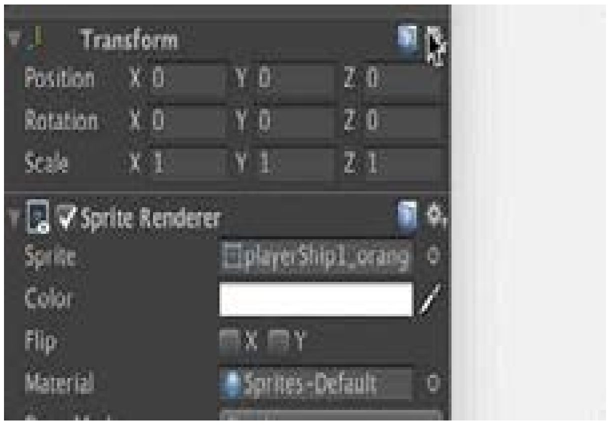
Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Figure 35: Transform component with default value

Try moving the ship around the Scene again. The X and Y numbers in the Position row of the Transform area should change as the ship moves. Enter the X value “8” and the Y value “4” in the Position row and the ship should move to the top right corner of the Scene. (If you’re having trouble finding the ship, you may have forgotten to zoom out in the last step. Select the Main

Camera and hit the F key while the cursor is hovering over the Scene view to zoom out and see your Main Camera's entire field of view.)

Let's reset our ship to the middle of the screen by clicking on the little gear icon at the top right corner of the Transform component, and selecting Reset from the list.



Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sprite Renderer
Sprite playerShip1_orang
Color
Flip X Y
Material Sprites-Default

Figure 36: Reset Transform on the Transform component

Now change the size of the ship by giving it new values in the Transform component's Scale line. Change the X to 2 and the Y to 2. What happens? Try different numbers and see how it affects the size of your ship.

When you change these numbers it makes changes in your game that you can see.

Reset the Transform once you are done.

Colliders

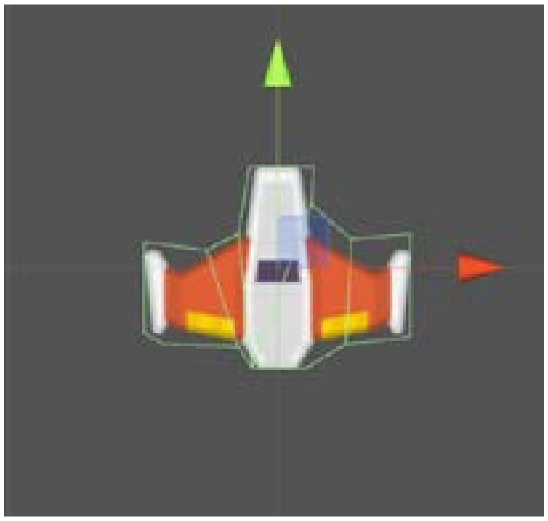


Figure 37: Ship with a green polygon collider around it

Next we're going to add a Collider component to our ship. A Collider is something that the computer uses to determine whether two objects are touching or not. This will allow our ship to interact with other GameObjects using Unity's built in physics engine.

Select **Ship** in the Hierarchy and click the Add Component button in the lower right corner of the screen.

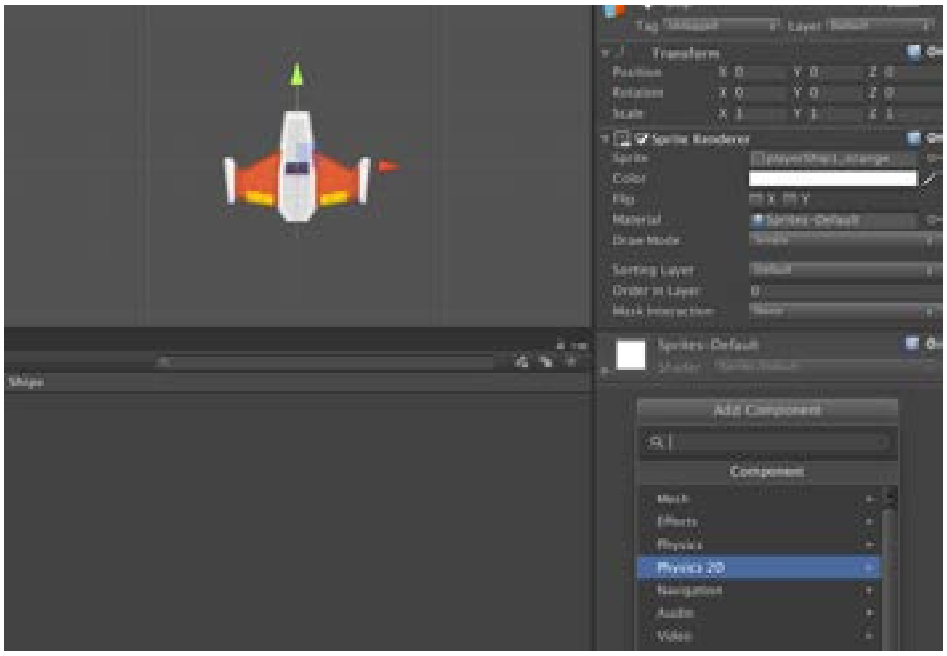


Figure 38: Physics 2D selected from the Add Component button

Select select Physics 2D and then choose Polygon Collider.

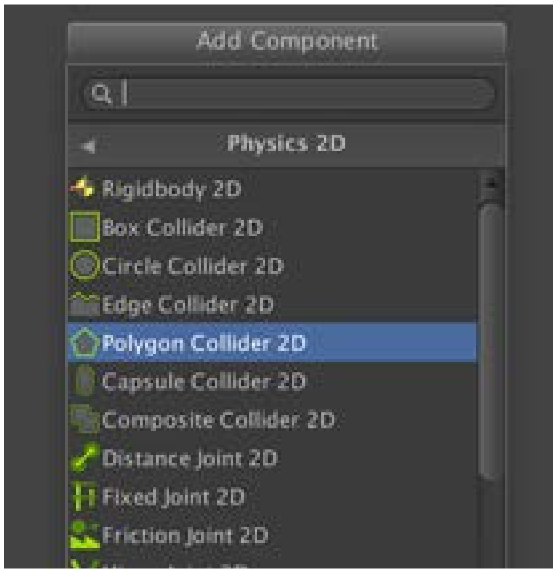
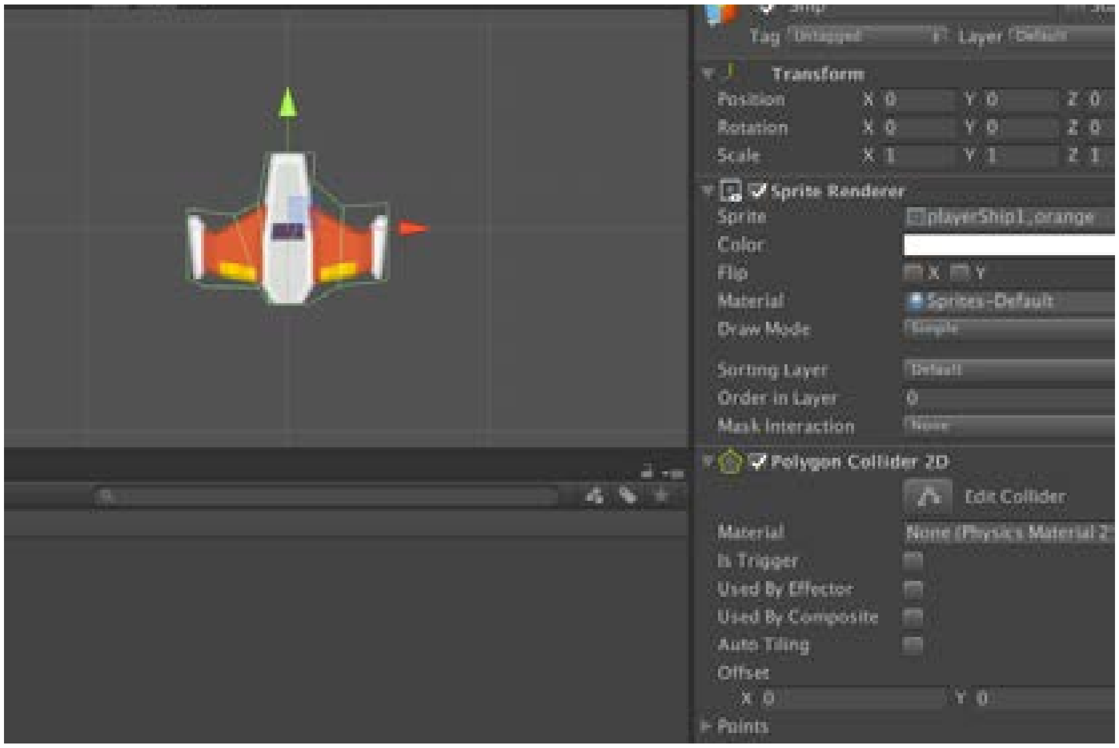


Figure 39: Polygon Collider 2D selected in the Add Component window

Now that you've added a Polygon Collider to Ship you can see the Collider as a green outline around the ship and a Polygon Collider 2D component has appeared in the Inspector panel.



Tag Untagged Layer Default

Transform

```
Position    X 0    Y 0    Z 0
Rotation    X 0    Y 0    Z 0
Scale       X 1    Y 1    Z 1
```

Sprite Renderer

```
Sprite             playerShip1_orange
Color
Flip               X  Y
Material           Sprites-Default
Draw Mode          Simple
Sorting Layer      Default
Order in Layer     0
Mask Interaction   None
```

Polygon Collider 2D

```
                   Edit Collider
Material           None (Physics Material 2
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset
  X 0              Y 0
Points
```

Figure 40: Polygon Collider component attached to the Ship GameObject. A Green outline surrounds the ship

Congratulations you've added a Polygon Collider 2D component. As we continue to work in Unity you may notice that you might add a lot of components to a single GameObject. You can minimize a component by clicking on the little triangle icon next to it's name or by clicking on the name of the component itself. You can also click again to make the component appear full sized. Try making the Polygon Collider 2D component small and then large again.

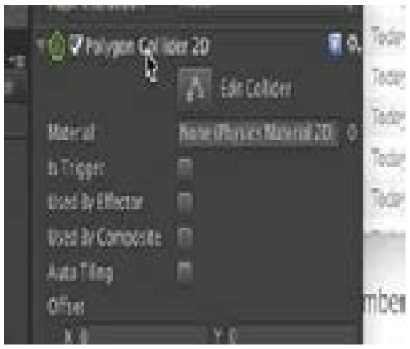


Figure 41: Minimizing and maximizing the Polygon Collider 2D component by clicking on it's name

Facilitator Note:

Unity automatically creates a Polygon Collider around a GameObject when you add it as a component, but you can edit the shape of the collider if you want to customize it. Just click the button that next to "Edit Collider" on the Polygon Collider 2D component and you should see several green dots around the Collider when in the Scene view. You can click and drag on those points to reshape your Collider, or click on on of the lines to add a new green dot.

Parent and Children GameObjects

Let's try some more advanced things with our GameObject.

GameObjects can be nested inside other GameObjects. Some are just empty containers that contain many GameObjects at the same time. Creating a nested tree of GameObjects in our Hierarchy allows us to do many advanced things in Unity that you can only do once GameObjects are grouped in this way.

Let's create an empty GameObject called **Player** to hold our ship GameObject and the other GameObjects we are going to create.

Click on the Create button on the top left of the Hierarchy window and select Create Empty

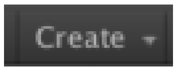


Figure 42: Create button of the Hierarchy

Right-click on the new GameObject rename it Player.

Now click on **Ship** and drag and drop it on top of **Player**. This makes **Player** the parent GameObject and **Ship** its child.

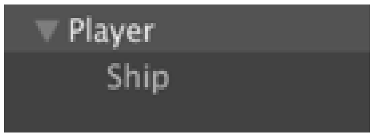


Figure 43: Hierarchy showing Ship as a child of Player

Making parent/child relationships between GameObjects can be very useful because child GameObjects will inherit properties from their parents. For example, if you have several different child GameObjects, and they all have the same parent, they will all move together if you move the parent. We will get further into these relationships when we start working with scripts later in the project.

Landing Legs

The player character we are making is a Lunar Lander for a Moon Landing game. The goal of our game is to navigate a treacherous terrain and land our vehicle in a specific location. We want to make it so that our ship will crash land if it plunges head first into the ground, but if we land gently on our landing legs we will be safe.

Go back to the Sprites folder in the Project window at the bottom and find the **Blocks** folder.

Inside that folder are several different materials folders. Choose from **Glass**, **Metal** or **Stone** and open one of those folders. Then add one of the rectangular or square shapes from the folder to the scene by adding it to the Hierarchy window. Make it a child of **Player** as well.

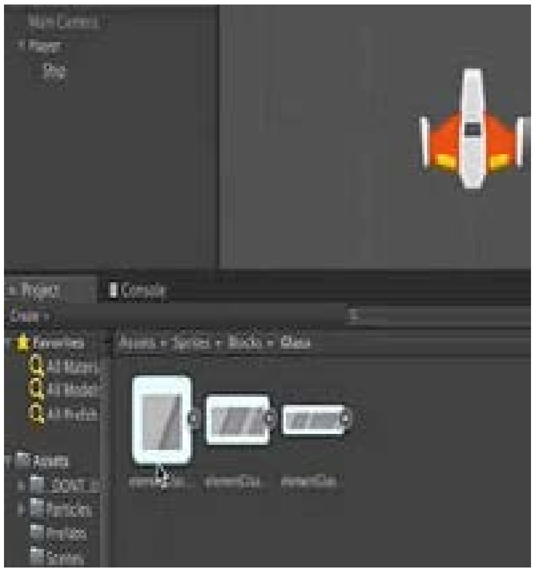


Figure 44: Dragging a Sprite from the Project window directly onto Player in the Hierarchy

Rename this new block "LeftLander" and play with its size and position until it looks like it could be the left landing leg of your ship. The shape should be tall and skinny and the position should be slightly below and to the left of the center of the ship.

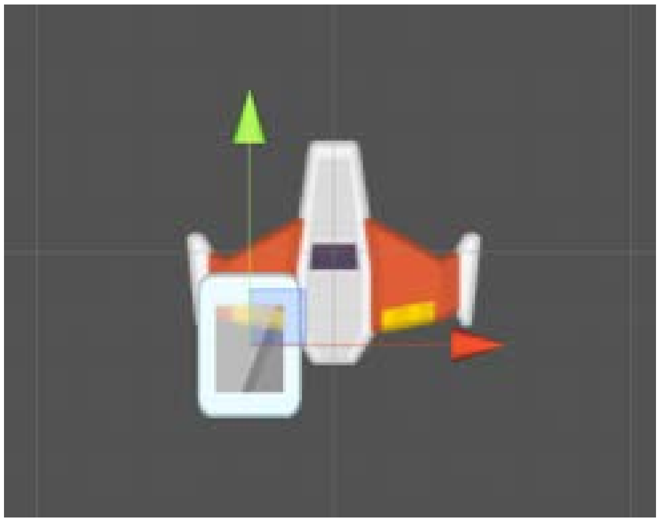


Figure 45: LeftLander in position in the lower left quadrant of the Ship

Select **LeftLander** and click on Add Component from the Inspector. Choose Physics 2D and then select Box Collider 2D to add a box Collider to the **LeftLander**. This is a simple Collider that perfectly fits our rectangular landing leg and uses very few system resources.

Select **LeftLander** in the Hierarchy and create a copy of it it by right clicking and selecting Duplicate. This will create a new identical version of **LeftLander** called "LeftLander (1)". Rename this duplicate "RightLander." Make sure that **RightLander** is also a child of **Player**.

Go to the Transform window for RightLander and remove the negative sign from the X position. This should make it snap into place on the other side of your ship. If not, manually move your leg until it looks something like this:

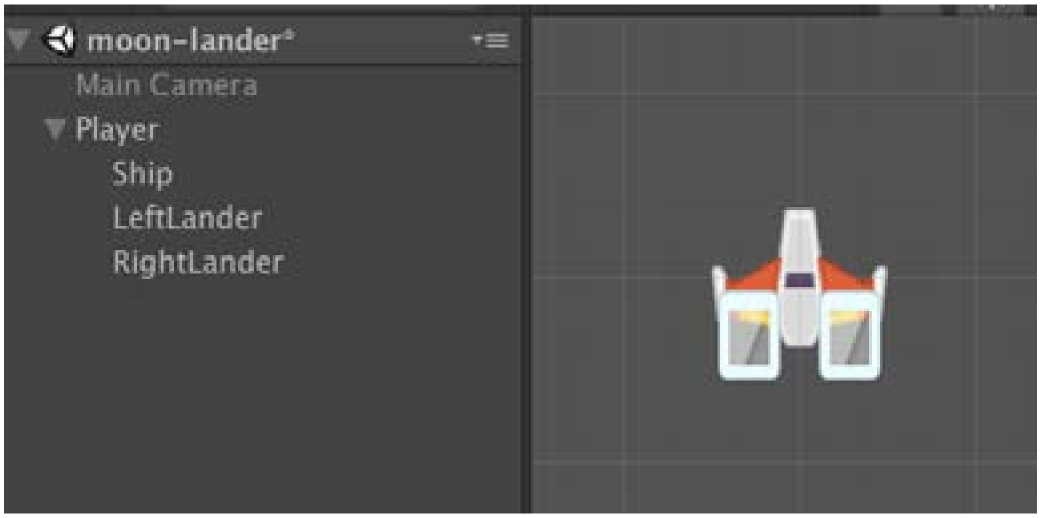


Figure 46: Hierarchy with LeftLander, RightLander and Ship as all children of Player as well

Moving the Player

In the Project window go back to the top level **Assets** folder, then find the Scripts folder, followed by the **Movement** folder.

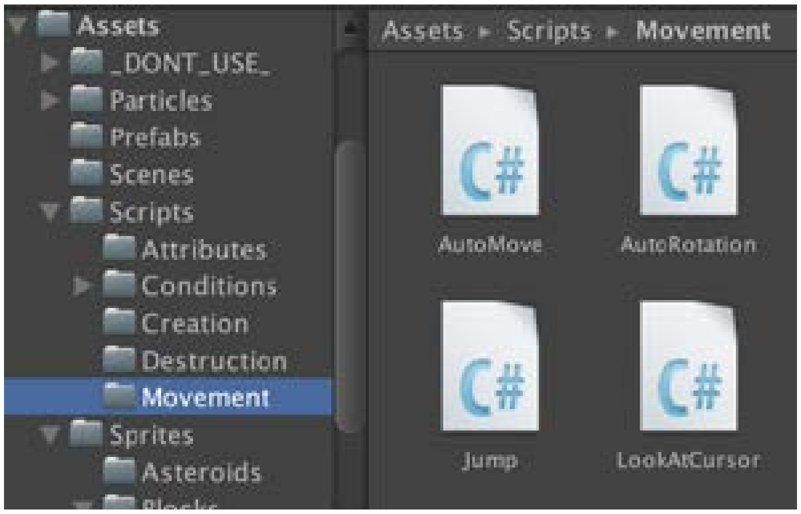


Figure 47: Project window with the ***Movement*** *folder selected and the drop-down menus on the left showing the full file path*

Select the **Player** GameObject (the parent one) so that it is in focus. You'll know you selected the Parent object because the ship and both of the landers will be outlined in green.

First we'll give the player the ability to push the ship forward by pressing the space key with the **PushWithButton** Script.

Drag the script "Push with Button" to the Inspector window while **Player** is in focus.

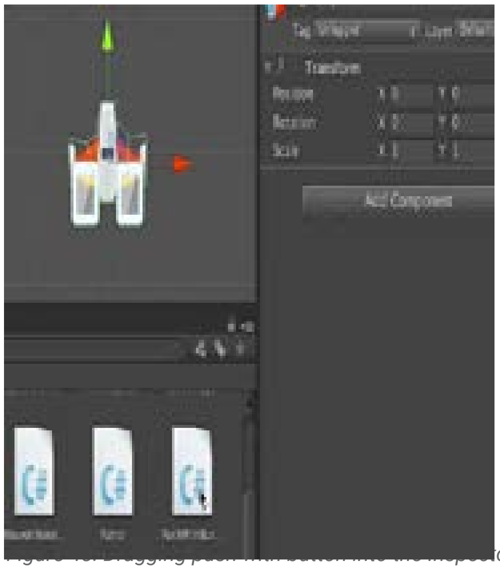
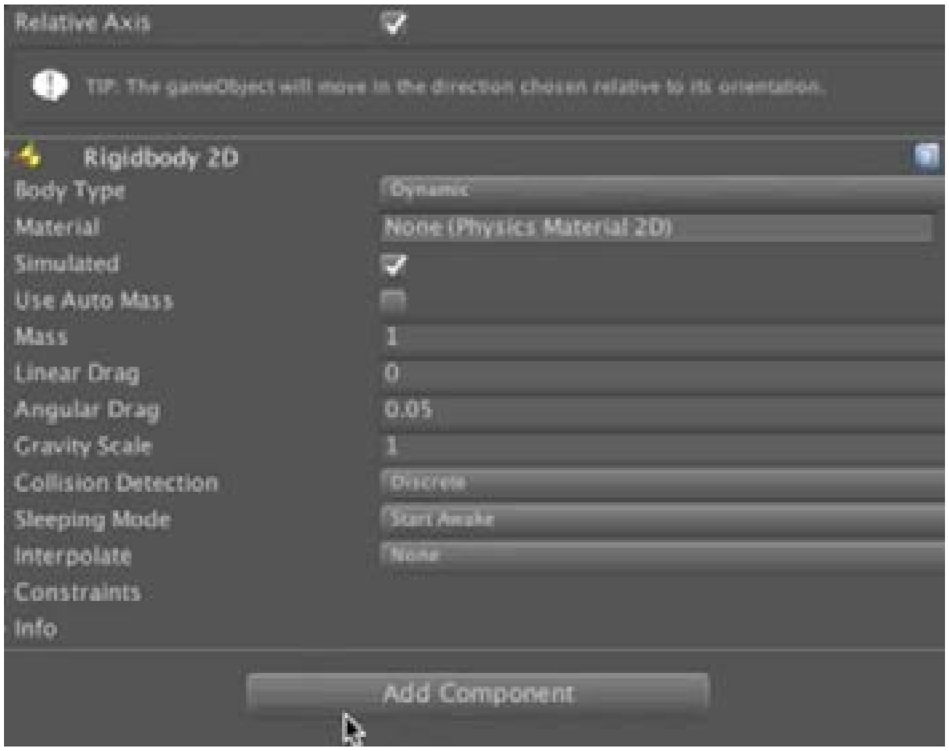


Figure 48: Dragging push with button into the Inspector

Once the game starts this script will cause the Player to move forward with a certain amount of force whenever the space key is pressed.

Next we'll make the ship rotate by pressing the arrow keys on the keyboard.

You can drag the script into the Inspector while the Player GameObject is still selected, or you can add the script from the Add Component button. To do this click Add Component and then select Scripts. Then choose **RotateWithArrows**



Relative Axis ✓

TIP: The gameObject will move in the direction chosen relative to its orientation.

Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	☐
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None
Constraints	
Info	

Figure 49: Clicking Add Component and selecting the script **RotateWitArrows**

This script makes it so that when you press the left and right arrows on your keyboard it will rotate the player either left or right.

Now reactivate the Main Camera by selecting it in the Hierarchy and then clicking on the check box next to its name in the Inspector.

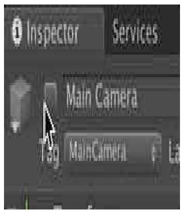


Figure 50: Main Camera in the Inspector

Now that we have a ship that moves and turns, we should be good to go, right?

Well if you enter Play Mode at this point you'll see that your ship just falls out of the sky. Give it a try: Click the triangle icon in the center of the top of the screen to enter Play Mode (it looks like a Play button).

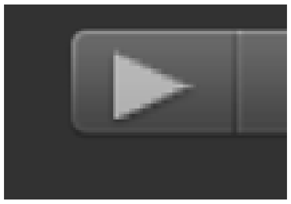


Figure 51: Play mode button

The reason it falls is because there is we have too much Gravity. We'll fix that in our next step.

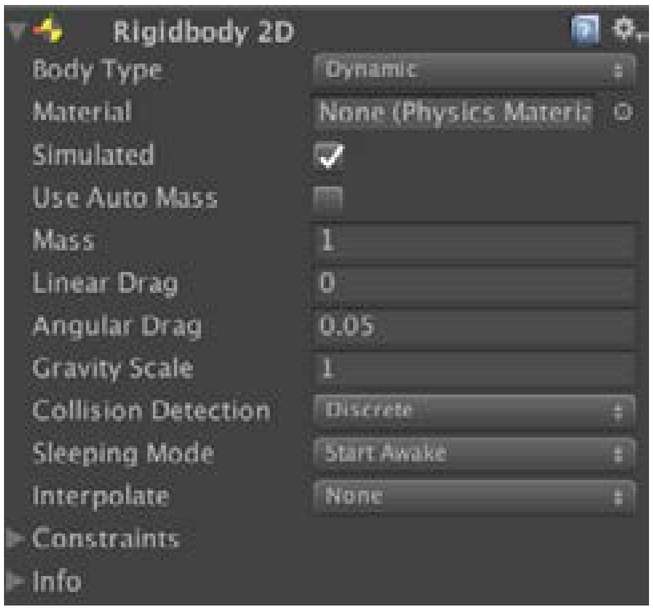
To leave Play Mode press the button again.

Make sure you don't accidentally stay in Play Mode since any edits you make to your game will not be saved while you are in Play Mode!

Gravity, Force and Drag

You may notice our ship just drops out of the screen when you enter Play Mode, and spins uncontrollably if you try to move. In order to fix this and make our game playable, we need to add some physics to it. We'll do that in this step.

Make sure you are no longer in Play Mode and find the Rigidbody 2D component of **Player** in the Inspector on the right. This was automatically added when we connected the **PushWithButton** script to **Player**.



Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Materia
Simulated	✓
Use Auto Mass	
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None
Constraints	
Info	

Figure 52: Rigidbody component

Set Gravity Scale to 0 and enter Play Mode again. This time you won't fall at all. Try moving and turning. You may now notice that your ship spins in circles when you press the left or right arrows and flies off uncontrollably when you hit the spacebar. This is because there is not enough drag on the ship.

Exit Play Mode and go back to the Rigidbody 2D Component of Player. There are number of different settings here. You should take some time to play with different numbers in each setting to see what they do.

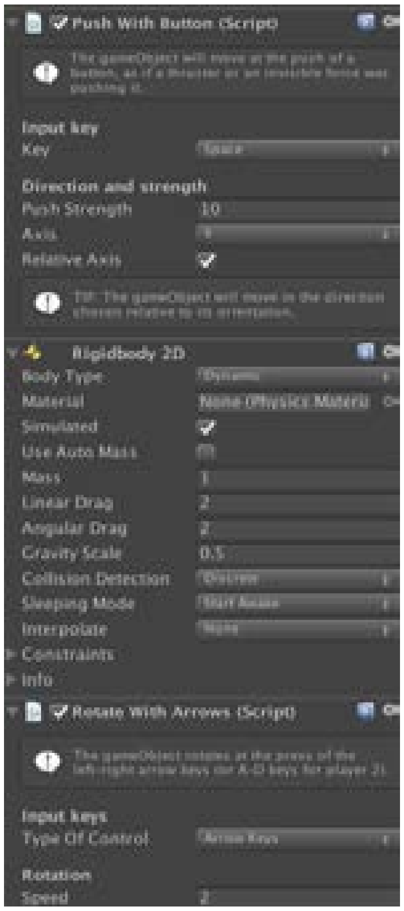
- **Linear Drag** controls how quickly something slows down as it moves across the screen. More drag will make it come to a stop faster, 0 drag will make it move forever with just a single push.
- **Angular Drag** controls how quickly something stops turning. It's just like Linear Drag but it controls the spinning of an object. More liner drag means the Player will stop turning after you let go of an arrow key , 0 drag will make it spin forever after pressing a direction key once.

- **Gravity** is gravity! More gravity will make an object fall faster. 0 Gravity and it will never fall.
- The **Push Strength** of PushWithButton controls how fast the Player goes when you hit the spacebar.
- The **speed** of RotateWithArrows controls how much the Player rotates after you press either the left or right arrow.

If you just want to set your settings to the recommended numbers for the Moon Lander game set everything to the following:

Mass to 1
Linear Drag to 2
Angular Drag to 2 (If you are having a really hard time steering you can try turning this to 4 for easy mode)

Gravity to .5
Push with Button Push Strength to 10
Rotate with Arrows speed to 2



```
Push With Button (Script)
  The gameObject will move at the push of a
  button, as if a thruster or an invisible force was
  pushing it.
Input key
Key                     Space
Direction and strength
Push Strength           10
Axis                    Y
Relative Axis           ✓
  TIP: The gameObject will move in the direction
  chosen relative to its orientation.
Rigidbody 2D
Body Type               Dynamic
Material                None (Physics Materi
Simulated               ✓
Use Auto Mass
Mass                    1
Linear Drag             2
Angular Drag            2
Gravity Scale           0.5
Collision Detection     Discrete
Sleeping Mode           Start Awake
Interpolate             None
Constraints
Info
Rotate With Arrows (Script)
  The gameObject rotates at the press of the
  left-right arrow keys (or A-D keys for player 2).
Input keys
Type Of Control         Arrow Keys
Rotation
Speed                   2
```

Figure 53: Recommended settings for Push With Button, Rigidbody 2D and Rotate With Arrows of the Player GameObject

Feel free to experiment with these numbers to give your game the feel that you want it to have. Enter Play Mode and test out your settings until they are to your liking. Then exit Play Mode.

Make sure you save your scene.

Level Design

Now that we're finished making our Player it's time to create our own unique level for our game. In Game Design, the art of creating individual levels is called Level Design and is an important part of the game making process.

Select the Move tool from the upper left corner of the screen.

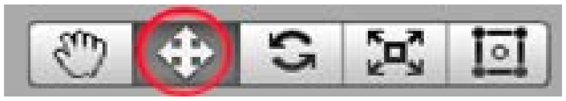


Figure 54: Hotkeys with a red circle around the Move tool

With the Move tool selected, click on one of the asteroids on the right side of the screen.

Remember: you can click and drag on the red and green arrows that appear behind the object to move it in a single direction, or click on the blue square and drag to move it freehand around the scene. Make sure you are clicking directly on the blue square or the colored arrows, otherwise your asteroid will not move.

Reposition the asteroids however you like, using the Move tool to place them around the game. Make sure there is a large enough path between asteroids that the player can move through in order to get to the End platform.

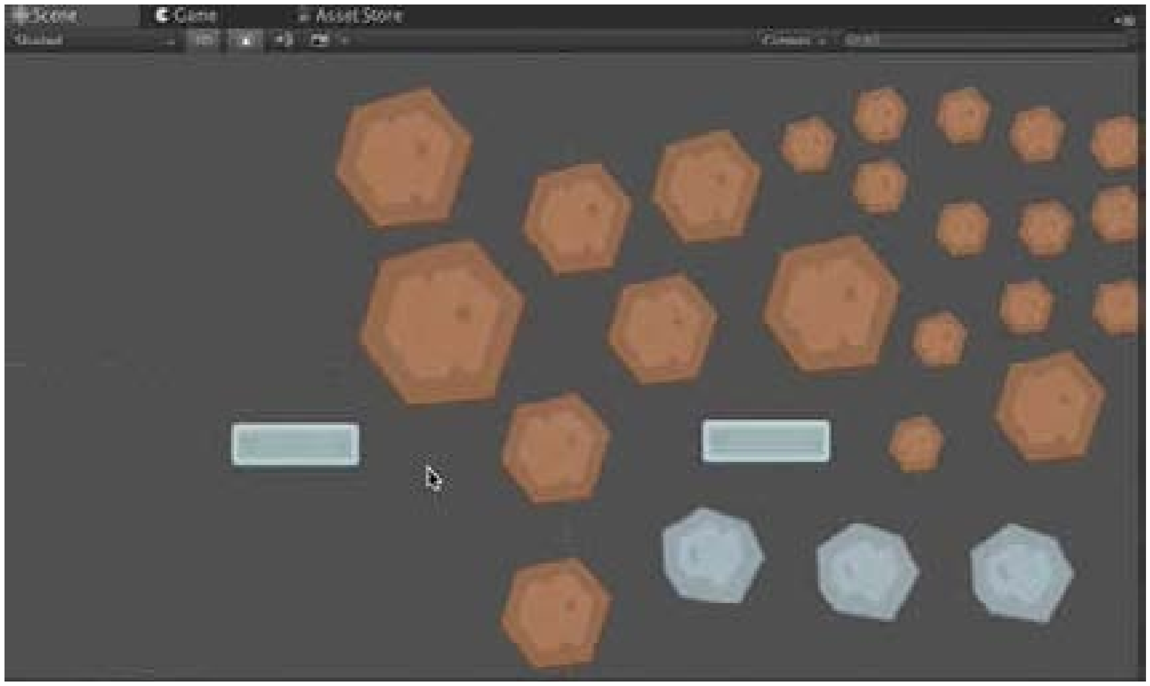


Figure 55: The Move tool Clicking on an Asteroid, Selecting the Blue Square and Moving the Asteroid to the left, then selecting a new Asteroid, clicking and dragging on the the red horizontal arrow of the selected object, selecting another Asteroid and moving it by the vertical axis

If you want to make an easy version of this level, place the platforms close together and put most of the asteroids near the bottom of the screen. If you want to make the level harder, place the platforms further apart from each other at different heights and place lots of asteroids in between them.

Enter Play Mode again to test your level. Don't forget to reactivate your Main Camera and any other GameObjects you deactivated while setting up your game. You can reactivate the Main Camera by selecting it in the Hierarchy so it is in focus, and the check the empty box next to the name "Main Camera" in the Inspector

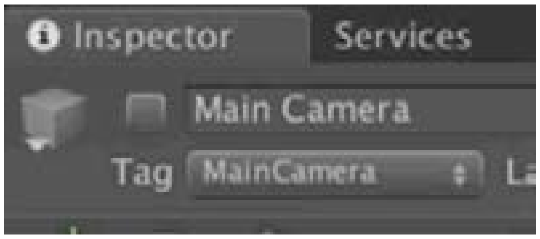


Figure 56: Activating and Deactivating the Main Camera in the Inspector

Facilitator Note:

If you notice that the words "Player 1 Wins!" appear when you start the game deactivate the WinPanel GameObject that is a child of UserInterface.

Test your game by navigating around the asteroids to get to the End platform. After playing for a few minutes ask yourself if your new level is too hard or too easy.

Exit Play Mode and adjust the position of your asteroids to give your level more or less challenge.

Once you have edited the game to create a level that has the amount of challenge you want, you are done. Congratulations, you fixed it and created your own new level!

Player Prefab

We've created an awesome player character in this level, and we're going to want to use this character again in the next Unity activity. In order to do this without having to make the Player all over again, we will create a Prefab, which is a type of asset that lives in the Project view and can be added to many different scenes. We will use the Player Prefab as our main character in the next activity, Unity Lander 2.

Select the Parent level **Player** GameObject in the Hierarchy. Click and drag it into the **Prefab** folder in the Project window.

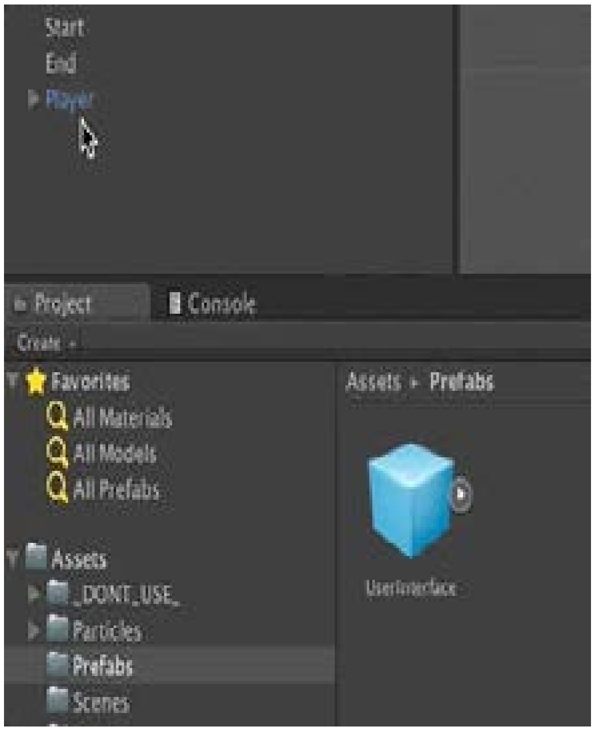


Figure 57: Dragging the Player GameObject into the Prefab folder in the Project Window

Once your Prefab is in the **Prefab** folder you can use it in any other scenes in this project. We'll come back to this in Unity Lander 2.

Save the Scene

Now that you've built your own new level it's time to save it, by saving the scene.

Go to the File menu and select "Save Scene as." Then name your scene with your name.

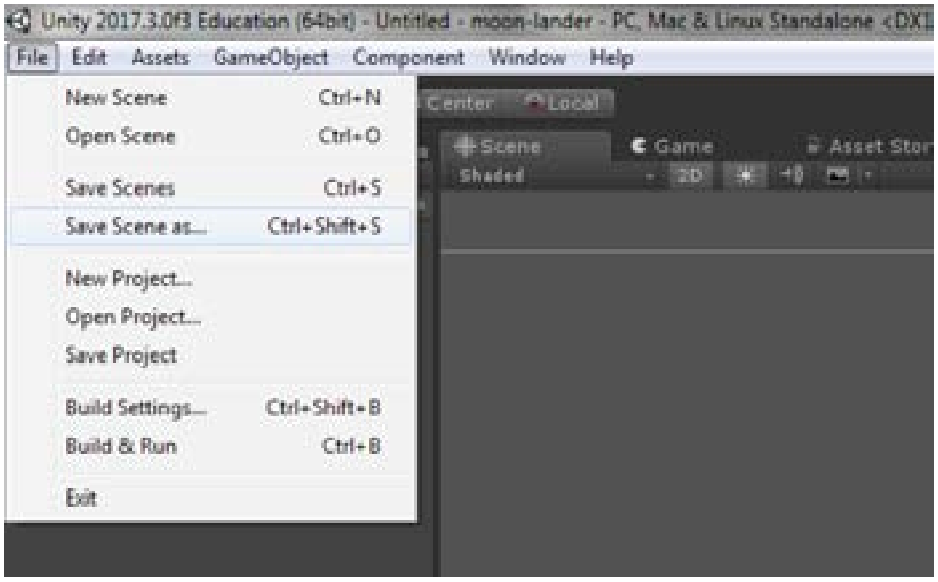

Unity 2017.3.0f3 Education (64bit) - Untitled - moon-lander - PC, Mac & Linux Standalone <DX1
File Edit Assets GameObject Component Window Help
New Scene Ctrl+N
Open Scene Ctrl+O
Save Scenes Ctrl+S
Save Scene as... Ctrl+Shift+S
New Project...
Open Project...
Save Project
Build Settings... Ctrl+Shift+B
Build & Run Ctrl+B
Exit
Center
Local
Scene
Game
Asset Stor
Shaded
2D

Figure 58: File Menu open and Save Scene as selected

Now you should see your new level in the top-level Assets folder. Drag and drop it into the **Scenes** folder alongside **level1** and the original version of **remix-me**.

Now we have installed Unity and signed into our account, imported Assets and loaded and played a Moon Lander game. You added GameObjects to the Hierarchy, added 2D Collider components and scripts to them, created Parent/Child relationships between GameObjects. After creating your Moon Lander character you moved GameObjects around the Scene view to fix a broken game and created a new custom level.

Try the next project, Unity Lander 2 to take the Moon Lander player you created, put it in a new scene and build the rest of the Moon Lander game from scratch. You will create obstacles, start and ending platforms and use scripts to make your ship explode when it collides with asteroids or give you a "You Win" message when you land on the goal platform, using the Assets you added in this project.

Feedback

We’d like to hear from you! Please take 5 minutes to fill out our survey!