**Lab 2: Setting up a VR Diorama**

# Technical Requirements

To implement the projects and exercises in these labs, you will need the following:

* A PC or Mac with Unity 2021.3.18 LTS or later, the XR Plugin for your device,
* Note: An Apple Silicon Mac will only work with Unity 2021.3.18

# Creating a New Unity Project

Let's create a new Unity project named VR\_is\_Awesome, or whatever you'd like:

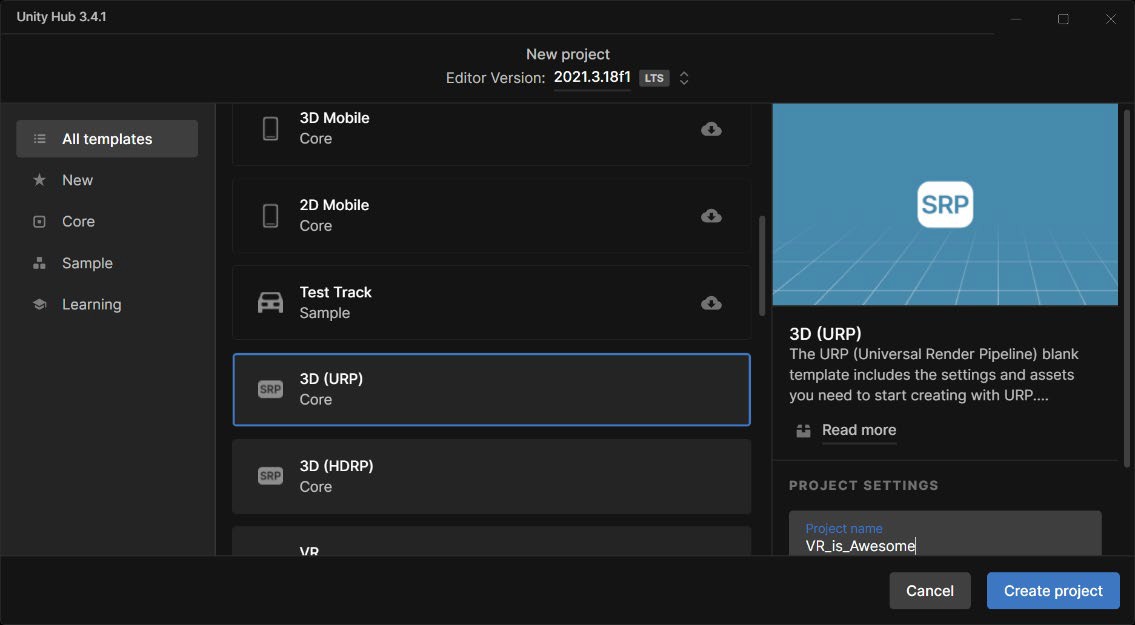
1. In **Unity Hub**, choose **Projects** in the left-hand menu tabs.
2. Select the **NEW** button to start a new project (note that the **NEW** button includes a dropdown list for selecting Unity version 2021.3.18.
3. In the **Create New Project** dialog box, you will be prompted to choose a **Template**.
4. For the projects in this lab, we will use the **Universal Render Pipeline**. Select

## Universal Render Pipeline.

Enter a project name, such as VR\_is\_Awesome. You can also select its location in your filesystem.

1. Select **CREATE** to create a new empty project and open the Unity Editor on your desktop.

A **Create New Project** dialog box is shown in the following screenshot:



Unity Hub gives you the option to choose a template when creating a new Unity project. The projects in this module will use the Universal Render Pipeline (formerly called the Lightweight Render Pipeline). If you are aiming for high-end graphics quality, you may decide to use the **High-Definition Render Pipeline** (**HDRP**) in your own projects instead

# Installing Additional Packages and Assets

Once you create and open a specific Unity project, there may be more packages and assets that you want or need to install. Unity provides several avenues for extending its features, platform support, editor tools, and runtime assets:

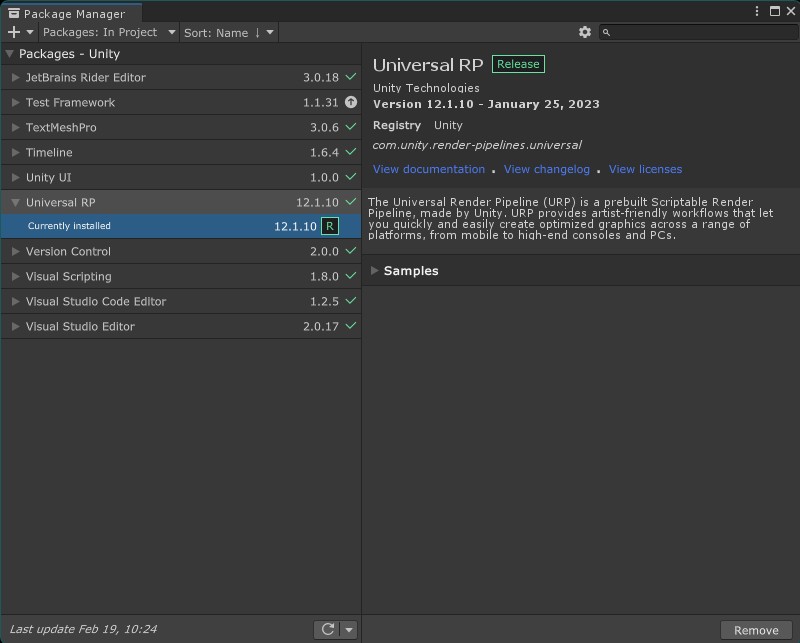
* + Package Manager: Use the Unity Package Manager to view which packages are available for installation or are already installed in your Project.
  + Import Unity Package: You can import asset package files (filename extension .unitypackage) from the menu **Assets | Import Package**
  + Import Asset: You can import individual assets that are created with another app or downloaded from the web with **Assets | Import New Asset**, or simply drag and drop a file into the Project window in Unity.

We'll use all of these mechanisms at one time or another later in this lab and throughout this module.

With a new project, it's always a good idea to open the **Package Manager** and review what's already installed in your project, so let's do that now:

1. Select **Window | Package Manager**.
2. By default, the filter is set to **In Project**.
3. You may notice that from time to time, newer updates to your packages will become available. With appropriate caution, you should update. Especially with point releases, these updates should only be bug fixes rather than potentially breaking features.
4. To add a package, you need to find uninstalled packages. Change the filter dropdown to **All Packages** to see all packages, including those not yet installed in your project.

The following image shows a **Package Manager** window where several packages have updates available:



# Creating a Simple Diorama

We will add a few objects to the scene, including a unit cube, a flat plane, a red ball, and a photographic backdrop.

1. To begin, let's create a new scene and name it Diorama:
2. If you haven't already, start a new scene in your opened project by selecting **File |** New Scene.
3. Then select **File | Save As...**.
4. Open the Assets/Scenes/ folder.
5. Set the file name to Diorama.
6. Press **Save**.

# Adding a Cube and a Plane

Let's add the first object to the scene, a unit-sized cube:

1. Within the **Hierarchy** window, click on the **+** (create) menu in the upper-left corner.
2. Click on **3D Object | Cube**.

A default cube will be added to the scene, centered on the ground plane at the (0, 0, 0) position, with no rotation and a scale of 1, as you can see in the **Inspector** window. The **Transform** component should have **Position** (0, 0, 0), **Rotation** (0, 0, 0), and **Scale** (1, 1, 1). These are the reset settings. If for some reason your cube has other **Transform** values, set them in the **Inspector** panel, or go through the following steps:

1. Click the three-dot menu icon in the upper right-hand side of the **Inspector**

window's **Transform** component.

1. Select **Reset** to reset the **Transform** values.

This cube has the dimensions of one unit on each side. As we'll see later, one unit in Unity corresponds to one meter in world coordinates. Its local center is the center of the cube.

Now, let's add a ground plane object to the scene:

1. In the **Hierarchy** window, click on the + (create) menu in the upper-left (or main

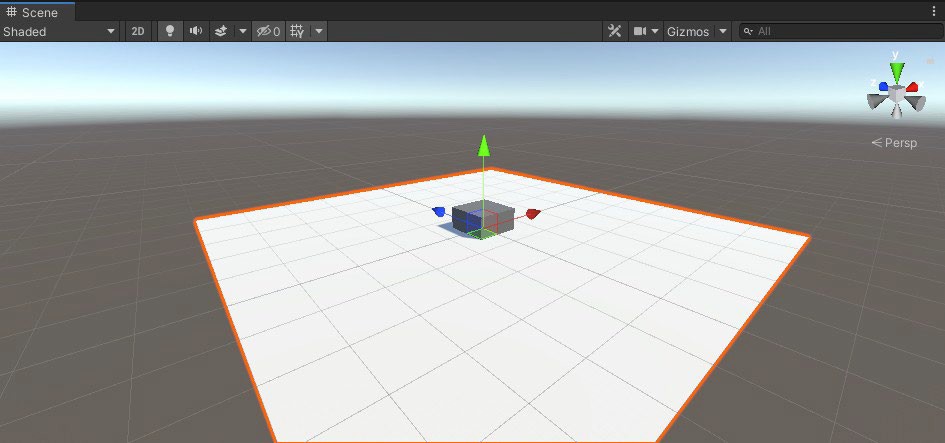
**GameObject** menu).

1. Select **3D Object | Plane**.
2. If necessary, reset its **Transform** values.
3. A default plane is added to the scene, centered on the ground plane at **Position** (0, 0, 0). Rename it as GroundPlane in the name field at the top of its **Inspector**.

Note that at a scale of (1, 1, 1), Unity's plane object actually measures 10 by 10 units in *X* and

*Z*. In other words, the size of GroundPlane is 10 by 10 units and its transform's **Scale** is 1. The cube is centered at **Position** (0, 0, 0),

This is shown in the following screenshot.



The **Cube** is submerged in the **GroundPlane** because its local origin is at its geometric center—it measures 1 by 1 by 1 and its middle point is (0.5, 0.5, 0.5).

Move the cube as follows:

1. Move the **Cube** onto the surface of the **GroundPlane**—in the **Inspector** panel, set its **Y**

position to 0.5: **Position** (0, 0.5, 0).

1. Let's rotate the cube a bit around the *y-axis*. Enter 20 into its **Y** rotation: **Rotation** (0, 20, 0).

In the previous screen capture, our **GroundPlane** is white. Yours probably still has the default gray color. Let's fix that by creating a new **Material** for the plane, as follows:

1. In the **Project** window, open the Materials folder. (If there isn't one, select **Create | Folder**, and name it Materials).
2. With the **Materials** folder selected, click on **+ | Material** (or right-click in the **Project** folder) to create a new material and name it White Material.
3. In the **Inspector** window, click the white rectangle to the right of **Base Map**, which opens the **Color** window. Select white (hexadecimal #FFFFFF).
4. Drag the **White Material** from the **Project** window onto the **Ground Plane** in the **Scene**

window.

1. Also, for this lab, let's **Auto Generate** the environment lighting by selecting **Window | Rendering | Lighting.** Ensure that the **Scene** tab is selected at the top of the window and check the **Auto Generate** checkbox at the bottom of the window.
2. Save your work by clicking **File | Save** and continue building your scene.

# Adding a Red Ball

Add a sphere to the scene:

* 1. From the main menu, select **GameObject | 3D Object | Sphere**.
  2. If necessary, select **Reset** from the **Inspector** panel's **Transform** component

Like the **Cube**, the **Sphere** has a radius of 1.0, with its origin at the center. It's hard to see the

**Sphere** as it is embedded in the **Cube**. We need to move the **Sphere** position.

In the **Scene** window, with the **Sphere** selected, make sure that the **Move** tool is active (the second icon in the top-left icon toolbar) and use the arrows of the *x*-, *y*-, and *z*-axes to position it. I left mine at **Position** (3, 0.75, -1.75).

Make this **Sphere** into a red ball, with a new **Material**:

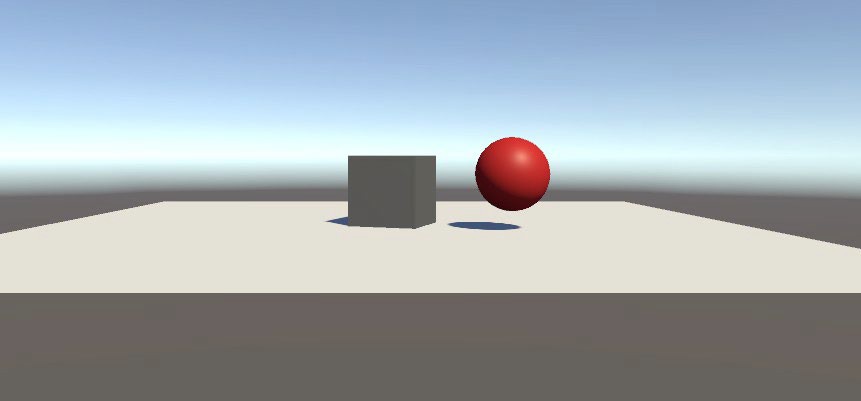
1. In the **Project** window, with the Materials/ folder selected, Click **Create | Material** and name it Red Material.
2. In the **Inspector** window, click the white rectangle to the right of **Base Map**, which opens the **Color** window. Choose red.
3. Drag the **Red Material** from the **Project** window onto the **Sphere** (note that you can drag it onto the **Sphere** in the **Scene** window, in the **Hierarchy**, or, provided the Sphere is currently selected, directly onto its **Inspector** window).

The **Sphere** should now look red. Save your scene.

The view in the **Scene** window is never necessarily aligned with the camera that renders the scene in the **Game** window

1. In the **Hierarchy**, select the **Main Camera**.
2. From the main menu, click **GameObject | Align View to Selected**

Here's what the scene looks like now:



# Making a Crate Material

You can Google search wooden crate texture to find one on your own:

1. In the **Project** window, created another **Asset** subfolder named Textures (by clicking on

## + | Folder).

1. Drag the crate image file from your filesystem into the Unity project's Textures folder. Ours is named crate.jpg.
2. In the **Project** window, in the Assets/Materials/ folder, create another material (right- click, then go to **Create | Material**) named Crate Material.
3. In **Inspector**, select the little doughnut icon next to the **Base Map (or** "Albedo"**)** to open a **Select Texture** dialog box, then choose the crate one (note that the search bar at the top of the dialog has the keyboard focus, so you can readily start typing the first few letters of the name to search).
4. Select the texture with a double-click (or single-click, then close the box with the red **X** in the upper right).
5. Set the **Base Map** to full color (white) by clicking its color chip and selecting white (#FFFFFF) so that the full fidelity of the texture pixels is not tinted.
6. Now, drag the **Crate Material** from the **Project** window onto the **Cube** in the **Scene** (or in the

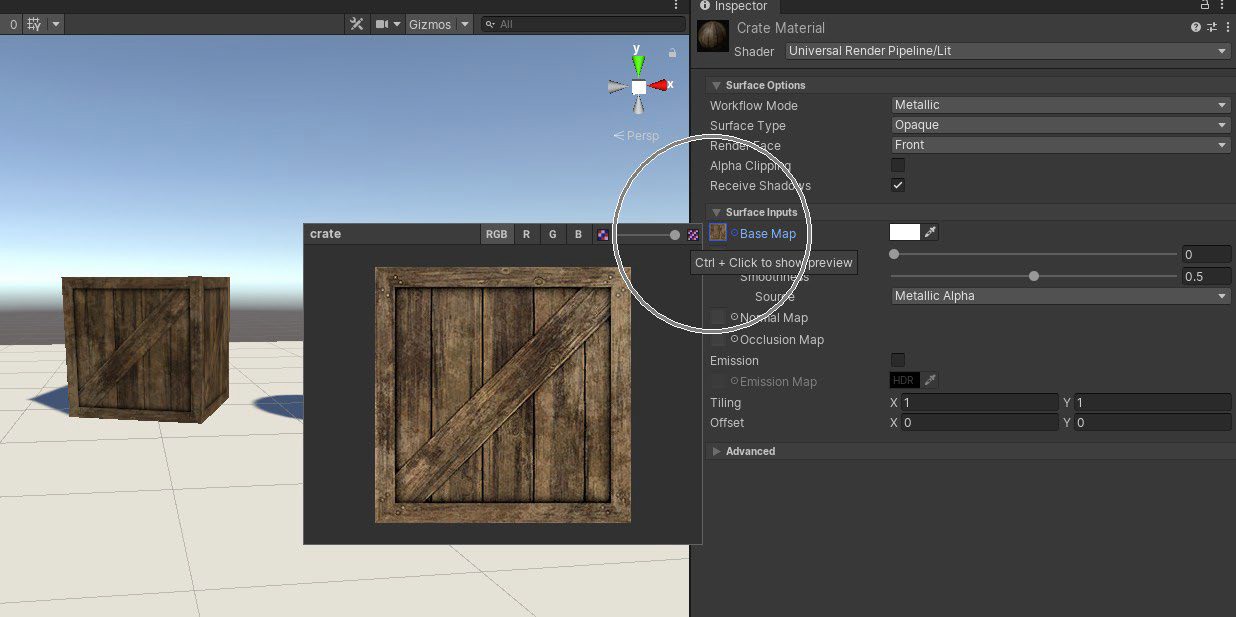
**Hierarchy** list).

1. Let's be more specific about our crate box. Rename it as Crate by selecting it in the **Scene** or

**Hierarchy** and editing its name at the top of the **Inspector**.

1. Save your work (**File | Save**).

The following screenshot shows the **Crate Material** in the **Inspector**, already applied to the cube.



Add another texture to the scene, this time as a backdrop image.

# Adding a Photo

Add a photo of the big-screen backdrop of our diorama.

The idea is to use a vista image with no nearby objects in the foreground. The steps are like the ones we used for the crate:

1. Drag the photo image from your filesystem into the Unity project's Textures/ folder. The provided one is named GrandCanyon.jpg.
2. Create a quadrilateral (a simple four-sided square) 3D primitive object by navigating to **GameObject | 3D Object | Quad** and naming it Photo.
3. In **Inspector**, reset its transform if necessary (**Transform | 3-dot-icon | Reset**).
4. Scale it to the size of the GroundPlane. Set its transform **Scale** to (10, 10, 10) (you may recall that the planes in Unity are 10 by 10 units, whereas other primitives, such as **Cube** and **Sphere**, are 1-unit sized).
5. Move it to the edge of the **GroundPlane** at a **Position** value of **Z** = 5, and above, at

**Position** value of **Y** = 5.

1. In the **Project** window, in the Materials/ folder, create another material (right- click and go

to **Create | Material**) named GrandCanyon Material.

1. rag the **GrandCanyon Material** from the **Project** window onto the **Photo** quad in the

**Scene** (or in the **Hierarchy** list).

1. In the **Inspector**, select the little doughnut icon next to the **Base Map** to open a **Select Texture** dialog box, then double-click to choose the **GrandCanyon** one.
2. Set the **Base Map** to full color (white) by clicking its color chip and selecting white (#FFFFFF).

In the scene, the image looks a little squished compared to the original. The original photo is 2576 x 1932 pixels or a 0.75 aspect ratio. But our quad is square. Let's fix that up:

1. Select the **Photo** quad in the scene.
2. Set its transform **Scale Y** to 7.5.
3. Set its transform **Position Y** to 3.75 (this is its new center offset, 7.5 / 2; in fact, you can type 7.5/2 into the *Y* value slot and Unity will evaluate the calculation).

Why 3.75? The height started at 10, so we scaled it to 7.5. The scaling of objects is relative to their origin. So now, half of the height is 3.75. We want to position the center of the backdrop

3.5 unit above the ground plane.

We have the size and position set up, but the photo looks washed out.

Disallow lighting effects on the photo using the **Unlit** shader. With **Photo** selected, change it to

**Unlit** by going through the following steps:

1. Select the **Photo** object.
2. In **Inspector**, note that the photo's **GrandCanyon Material** component in the **Inspector**

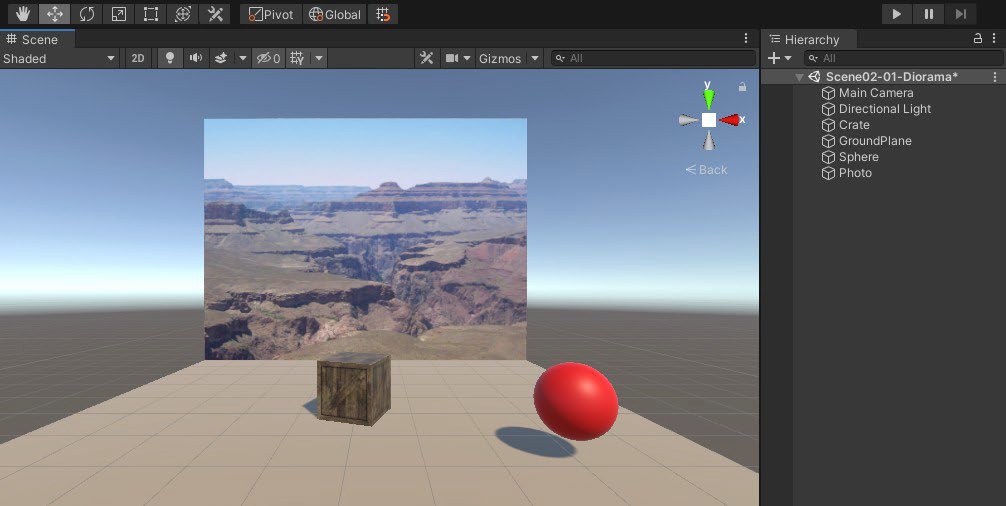
window has its default **Shader** set as **Universal Render Pipeline/Lit**.

1. Change the Material's **Shader** to **Universal Render Pipeline/Unlit** using the dropdown selection list.

Adjust the GroundPlane color to better match the photo ground:

1. Create a new material in the Materials folder and name it Ground Material.
2. Drag it onto the **GroundPlane** object.
3. Then, change its **Base Map** color. Use the dropper (icon) to pick a sandy tone from the image in your photo plane.

Here's what the scene looks like; yours should be similar:



Save your scene.

# Creating and Instantiating a Prefab

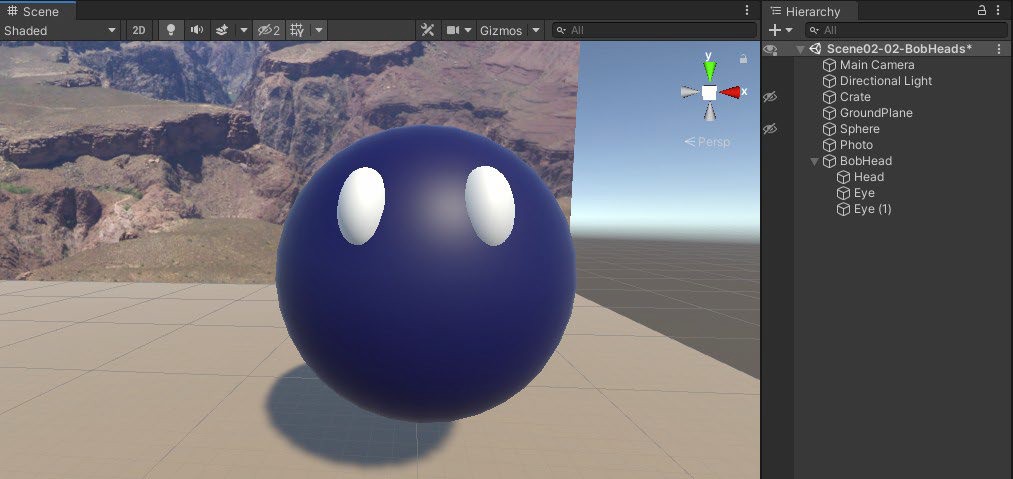
We're going to create a reusable character prefab named BobHead. First, let's build the little guy from a few primitive 3D objects, as follows:

1. In the **Hierarchy**, create an empty GameObject named BobHead (by clicking on **+| Create Empty**; name it BobHead).
2. Set its position to something like **Position** (3, 0.75, 1.25).
3. Create a child **Sphere** for its head (right-click the **BobHead** and click **3D Object | Sphere**). Name it Head.
4. In the **Project** Materials/ folder, create a new **Material**, named BobHead Material, and make it a dark blue (for example, #191963).
5. Drag the **BobHead Material** onto the **Head**.

Add the eyes:

1. Create a child white-colored eye (right-click the **BobHead**, then click **3D Object | Sphere**) and name it Eye.
2. Set its **Transform Position** (0.15, 0.2, -0.35) and **Scale** (0.15, 0.25, 0.25).
3. Drag the **White Material** that we created earlier from the **Project** Materials folder onto the **Eye**.
4. Duplicate the eye (select the **Eye** and press *Ctrl* + *D*).
5. Change its **Transform Position X** to -0.15.

The resulting model, including the scene **Hierarchy,** is shown in the following screenshot:



Now to create a prefab of the object. We'll save it in a project folder named

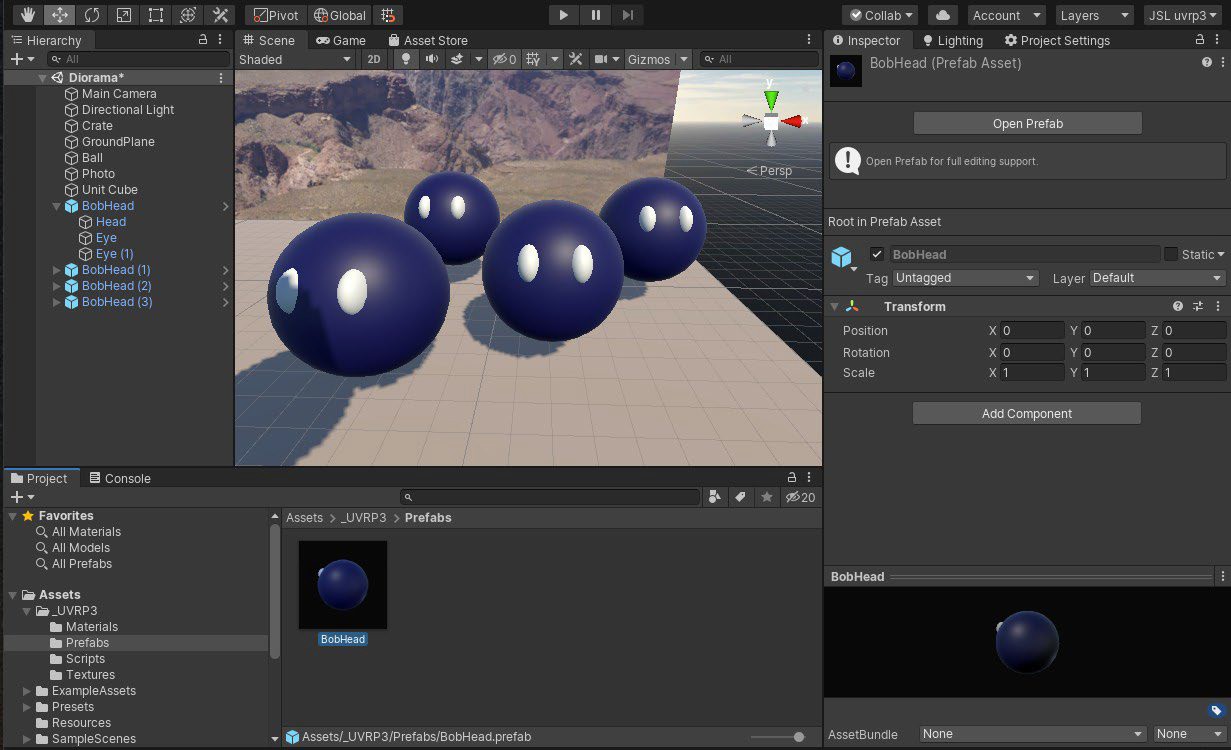
Assets/Prefabs/

* 1. In the **Project** window, create a new folder named Prefabs (at the root Assets/, right-click, then click **Create | Folder** and name it Prefabs).
  2. In the **Hierarchy** window, click and drag the **BobHead** GameObject into the **Project**

window's Prefabs/ folder we just created.

* 1. The **BobHead** prefab is now available as a template for other **BobHead** GameObjects.
  2. Add a few BobHead Prefabs to the scene.

The following screenshot shows the scene with multiple **BobHead** instances.



You can now modify the instance as you like, such as adjusting its position and scale.

# Editing and Overriding a Prefab

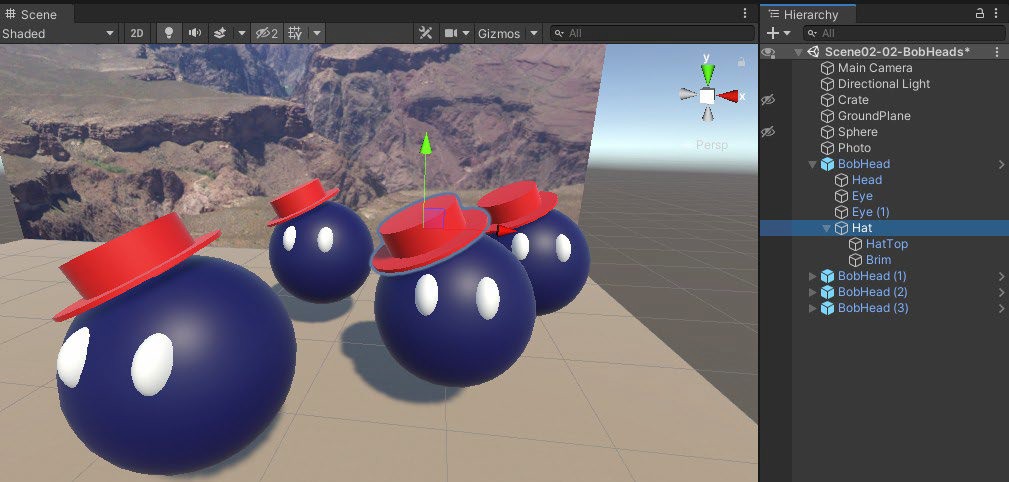
Select the **BobHead** prefab in the **Project** Prefabs/ folder and select **Open Prefab** to edit it Edit it first in the scene **Hierarchy**, then apply the overrides:

1. To add a hat, select one of the **BobHead** objects in the hierarchy, right-click, and select

**Create Empty**, then rename it Hat.

1. Set its **Position** to (0, 0.5, 0).
2. Right-click the **Hat** and select **3D Object | Cylinder** to add a cylinder as a child of Hat— name it HatTop.
3. Set its **Transform Scale** to (0.5, 0.1, 0.5).
4. Drag the **Red Material** from the Project's Materials/ folder (created earlier in this chapter) onto the **HatTop**.
5. Right-click the **Hat** again and select **3D Object | Cylinder** to add a cylinder as a child— name it Brim.
6. Set its **Transform Scale** to (0.75, 0.01, 0.75) and **Position** to (0, -0.075, 0).
7. Drag the **Red Material** from the **Project**'s Materials/ folder onto the **Brim**.
8. Adjust the **Hat** transform to **Rotation** (0, -15, 25) and **Position** (-0.2, 0.5, 0).
9. Select the **BobHead** object in **Hierarchy**
10. Click the **Overrides** in **Inspector**.
11. Press **Apply All**.

All the BobHeads in the scene have been updated, as shown in the following screenshot:



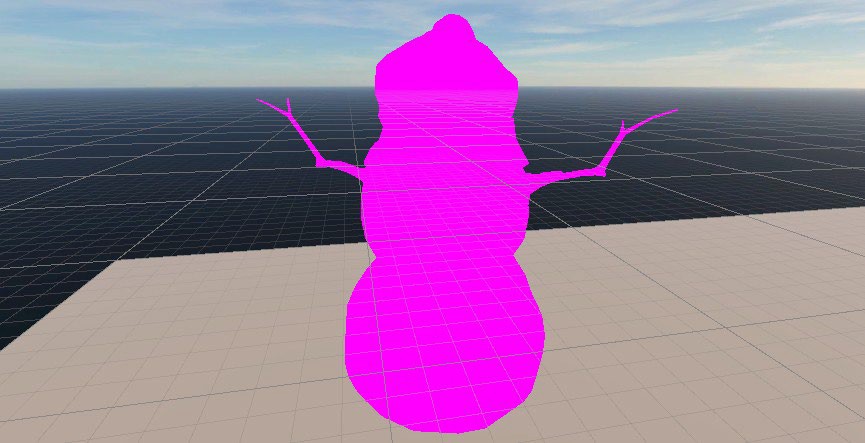
# Importing from the Unity Asset Store

A **skybox** is a panoramic texture drawn behind all objects in the scene to represent the sky or another vista at a far distance. Let's add a skybox to our project now:

1. In your browser, go to the **Asset Store** (<https://assetstore.unity.com/>) and search for Wispy Skybox, or go directly to [https://assetstore.unity.com/](https://assetstore.unity.com/packages/2d/textures-materials/sky/wispy-skybox-21737) [packages/2d/textures-](https://assetstore.unity.com/packages/2d/textures-materials/sky/wispy-skybox-21737) [materials/sky/wispy-skybox-21737.](https://assetstore.unity.com/packages/2d/textures-materials/sky/wispy-skybox-21737)
2. Click **Open in Unity**.
3. In Unity, the **Import Unity Package** dialog box will pop up, with all the contents selected. Click **Download** (if present), and then click **Import**.
4. Check your console window for errors (by going to **Window | General | Console**). I see none, so that's good.
5. Open the scene's **Lighting** window (by going to **Window | Rendering | Lighting**).
6. Select the **Environment** tab at the top of the window.
7. At the top is a slot for **Skybox Material**. Click the doughnut icon, which opens the **Select Material** dialog.
8. Search using the string wispy.
9. Select one of the skyboxes to use in the scene.

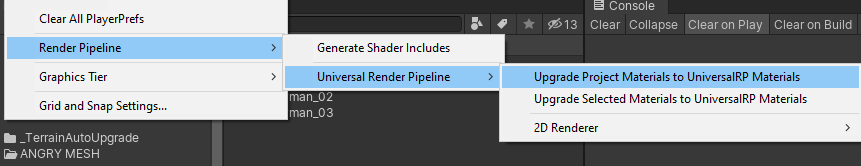
Import a model with a relatively high-quality PBR material. Add the *FREE Snowman* from Angry Mesh:

1. In your browser, go to the Asset Store (<https://assetstore.unity.com/>) and search for free snowman, or go directly to [https://assetstore.unity.com/](https://assetstore.unity.com/packages/3d/props/free-snowman-105123) [packages/3d/props/free-](https://assetstore.unity.com/packages/3d/props/free-snowman-105123) [snowman-105123.](https://assetstore.unity.com/packages/3d/props/free-snowman-105123)
2. Click **Open in Unity**.
3. In Unity, **Import** the package.
4. In the **Project** window, go to the Assets/ANGRY MESH/Snowman/Prefabs/folder.
5. Drag the **Snowman\_01** prefab into the scene.



Converting the imported materials to our RP:

Select **Edit | Render Pipeline | Universal Render Pipeline | Upgrade Project Materials to UniversalRP Materials**, then click **Proceed**:



# Using Unity Legacy Standard Assets

Several of the labs in this module use a variety of assets that Unity has Since Unity 2018, it was removed from the standard Unity install and became available on the Asset Store instead. We're still going to use this package, but we'll need to make some adjustments for compatibility with Unity 2021.3.18 or later and the Universal RP.

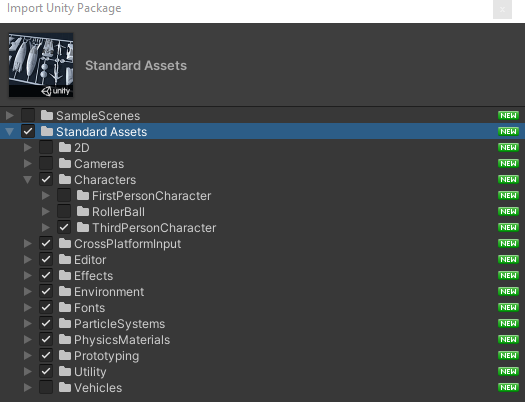
You can import the entire package or a subset as described in the following steps:

* 1. In your browser, go to the **Asset Store** (<https://assetstore.unity.com/>) and search for Unity Standard Assets, or go directly to [https://assetstore.](https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2017-3-32351) [unity.com/packages/essentials/asset-packs/standard-assets-for-unity-](https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2017-3-32351) [2017-3-32351.](https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2017-3-32351)

*If it is not available online use ‘StandardAssetsUsedInIMAT3002.unitypackage’ file in LZ and import it to your project: Assets in main menu/ Import Package/custom Package*

* 1. Click **Open in Unity,** click **Download,** and then click **Import**.
  2. In Unity, the **Import Unity Package** dialog box pops up, with all the contents selected.
  3. Uncheck the SampleScenes folder.
  4. Uncheck the Standard Assets/2D folder.
  5. Uncheck the Standard Assets/Cameras folder.
  6. Uncheck the Standard Assets/Characters/FirstPersonCharacter folder.
  7. Uncheck the Standard Assets/Characters/RollerBall folder.
  8. Uncheck the Standard Assets/Vehicles folder.

The import box with items that we don't need for this module is shown in the following image:



Import the assets, then correct any import errors as follows:

1. Click **Import**.
2. Check your **Console** window for errors by going to **Window | General | Console**.
3. There may be numerous import errors, but most of them will have been corrected by Unity. Click the **Clear** button
4. The remaining errors need to be resolved manually. Two script files have obsolete references. Delete these scripts.
   * Assets\Standard Assets\Utility\ForcedReset.cs
   * Assets\Standard Assets\Utility\SimpleActivatorMenu.cs
5. The import can now continue. More errors may be found and fixed by Unity. Click **Clear**

again.

That should complete the import. Next, let's attempt to convert materials to the current render pipeline:

Select **Edit | Render Pipeline | Universal Render Pipeline | Upgrade Project Materials to UniversalRP Materials** and then click **Proceed**.

You can browse the warning messages, but don't worry about them; we'll find incompatible materials soon enough.

One of the models that we plan to use soon is the Ethan humanoid model. Let's add him to the scene:

1. In the **Project** window, drill down to the Assets/Standard Assets/Characters/ThirdPersonCharacter/Prefabs/ folder.
2. Drag the **ThirdPersonController** into the **Scene**.
3. You may want to turn him around so that he's facing the camera by going to

**Transform | Rotation | Y**: 180.



Ethan has a third-person control script that lets you control his animation with user input. We will work on VR input devices in later chapters, so for now, let's try it out with just the keyboard:

1. Click on the **Play** icon at the top of the Unity window in the center to start your game.
2. Use the *W*, *A*, *S*, and *D* keys to move him around.
3. Click on the **Play** icon again to stop the game and return to edit mode.