CODE NINJAS®

Bronze Belt Ninja Guide

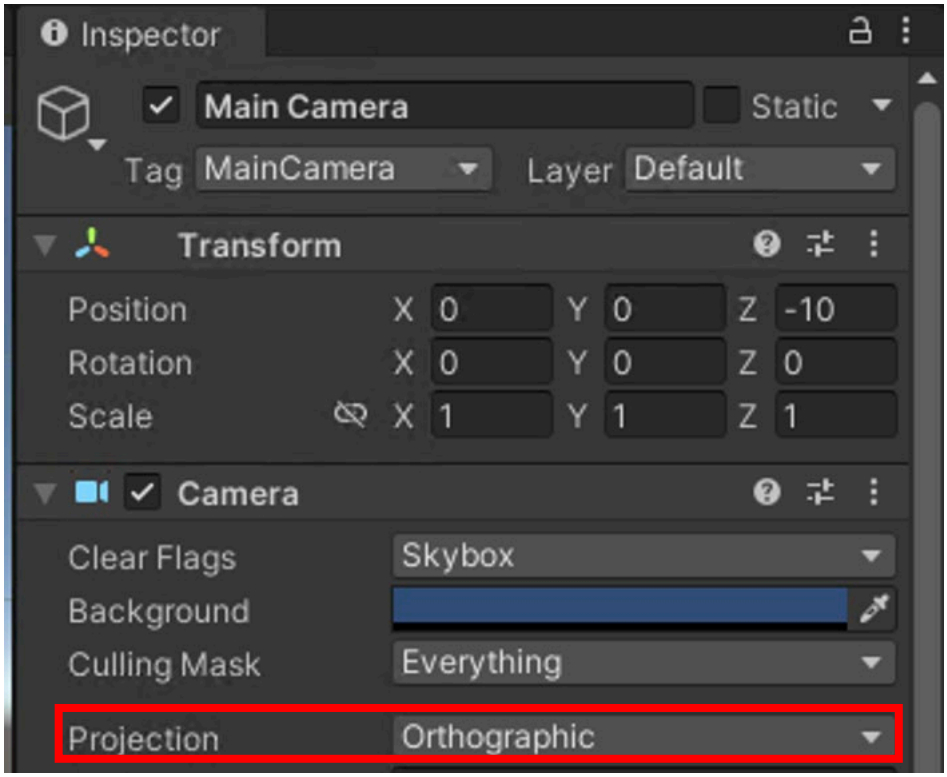
Activity 02: Scavenger Hunt

Activity 2: Scavenger Hunt

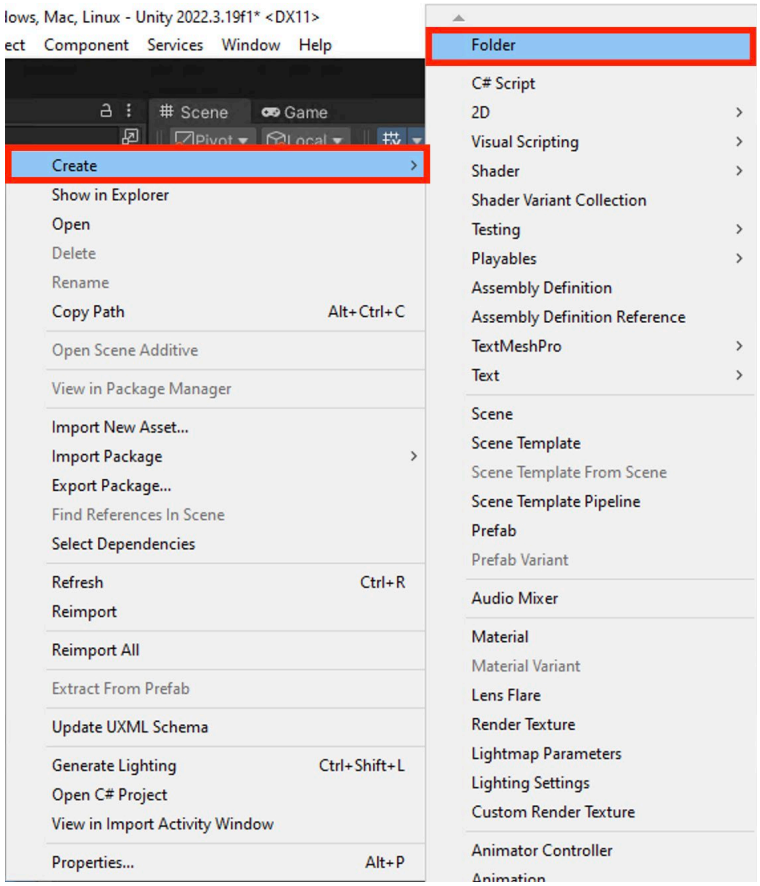
The previous activity was meant to serve as a quick introduction to working with Unity. This time, you'll build a simple platform game. To start, open Unity and create a **new** 2D project. Give it a name like **YourInitialsScavengerHunt**. For example, if your name was John Ninja, your project would be named **JNScavengerHunt**.

1 Depending on your layout settings, your display may not look exactly like this. You are free to work with the layout that you prefer. For this activity, we used the **Default** layout. Refer to the first two steps in the first activity if you are uncertain about how to change the layout.

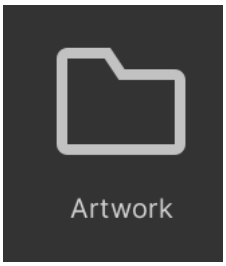
2 Just like the previous activity, we will be using the **2D** camera to edit our scene. Select the **Main Camera GameObject** in the **Hierarchy** panel. If the **Main Camera's Projection** is not already set to **Orthographic** then change the **Projection** from **Perspective** to **Orthographic** as shown below.



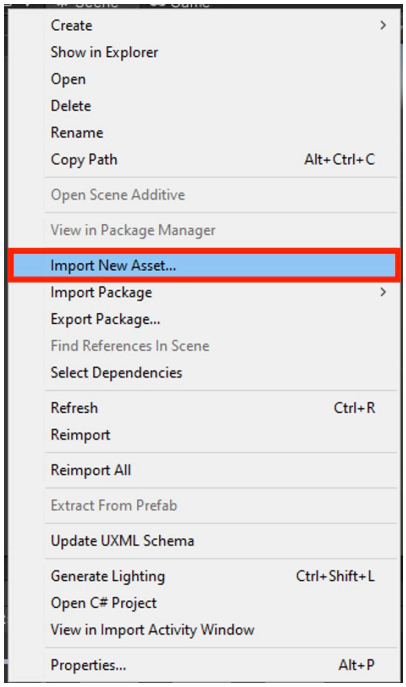
3 There will be some artwork provided for you to use in this game. Before adding it to the game, it's a good idea to have somewhere to put it. It's important to have different types of files in different folders in Unity so that you can find them quickly and stay organized. In the **Project** panel, right-click on the **Assets** folder window, click **Create** and select **Folder** to add a new folder to the **Assets** folder.



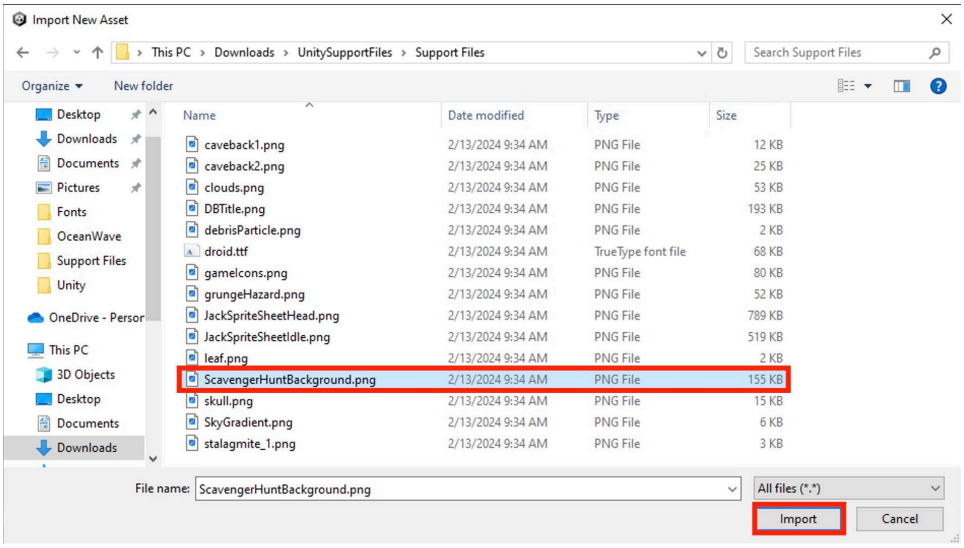
4 Rename the new folder to be "**Artwork**".



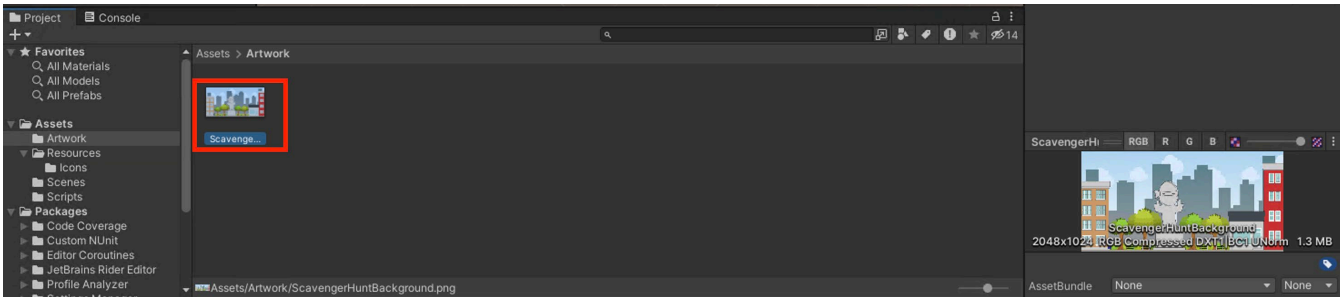
5 Open the **Artwork** folder. We are going to import an asset into this folder. There are many ways to import an asset. One way is to right-click in the folder that you want the asset and select **Import New Asset.**



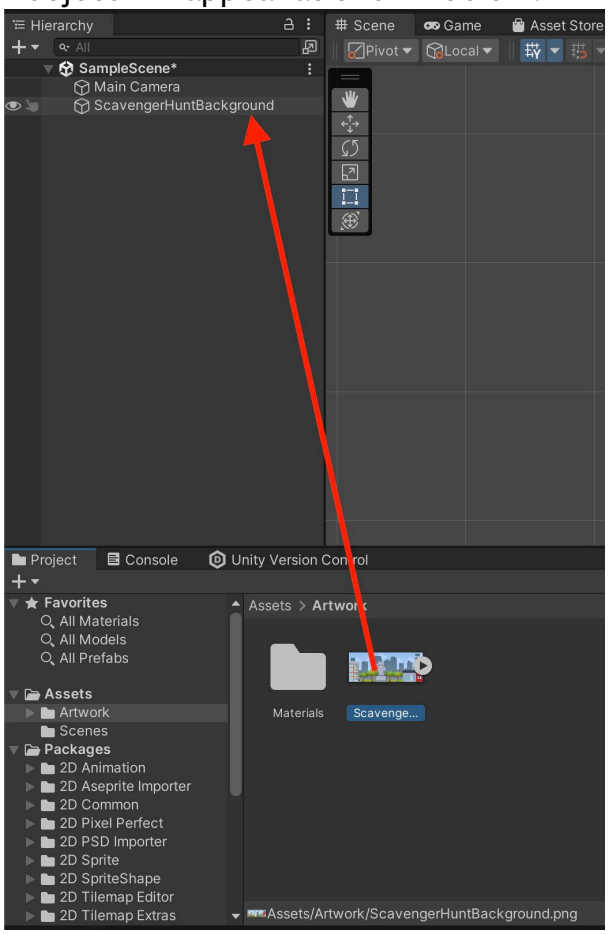
6 A new window will open with either File Explorer on Windows or Finder on Mac. Navigate to the folder where Bronze Belt assets are located, then click into the folder and select **Activity 02 - ScavengerHuntBackground.png** to import your new artwork.

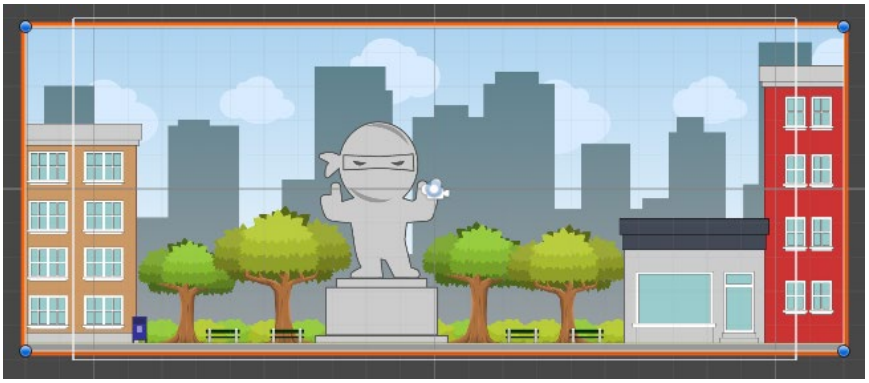


7 Once you select the import button, the new asset will now appear in the **Project** panel. Now that the asset is in the **Project** panel, select it to see all the information about it in the **Inspector** panel.



8 Now let's add the background to the hierarchy. Simply drag the background from the **Artwork** folder and a new object will appear as shown below.

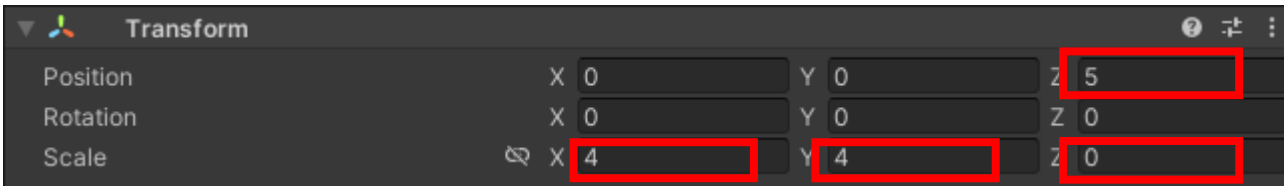




Pro Tip:

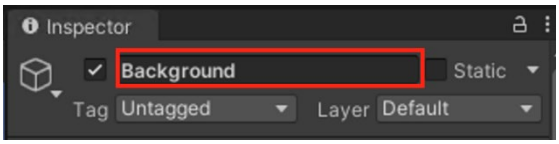
A **Quad** is short for quadrilateral. A **Quad** is like a plane in the fact that it only has four corners, but unlike a plane, it has depth. A **Quad** is ideal for holding simple graphic images as it only has two triangles in its mesh instead of the 200 found in the plane! Since you don't need all those extra triangles, why use them?

9 By default, the **Background** has a size of one unit. However, this is a little small for a background. Let's make it bigger by first selecting the background then increasing the scale by changing the x scale to **4** and the y scale to **4** in the Transform component as shown below. Change the z position to **5**.

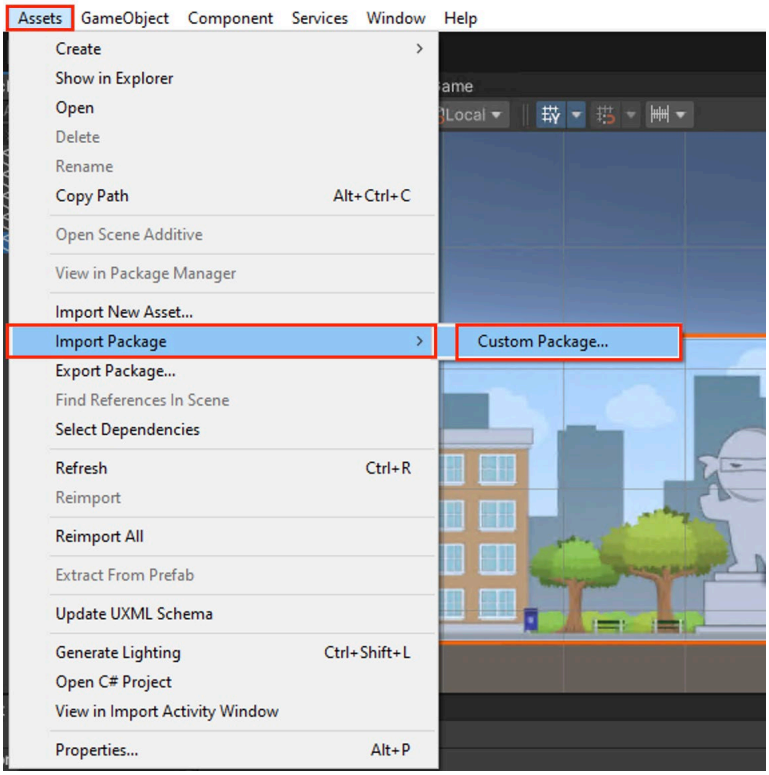


10 Ensure that in the inspector for **ScavengerHuntBackground** that the background is set as the sprite in sprite renderer.

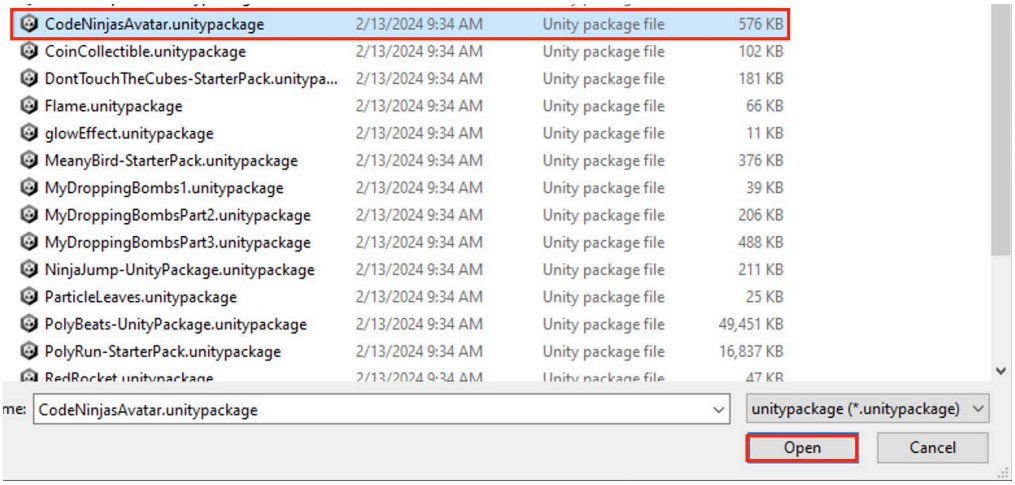
While the background is still selected, change the name from **ScavengerHuntBackground** to **Background**.



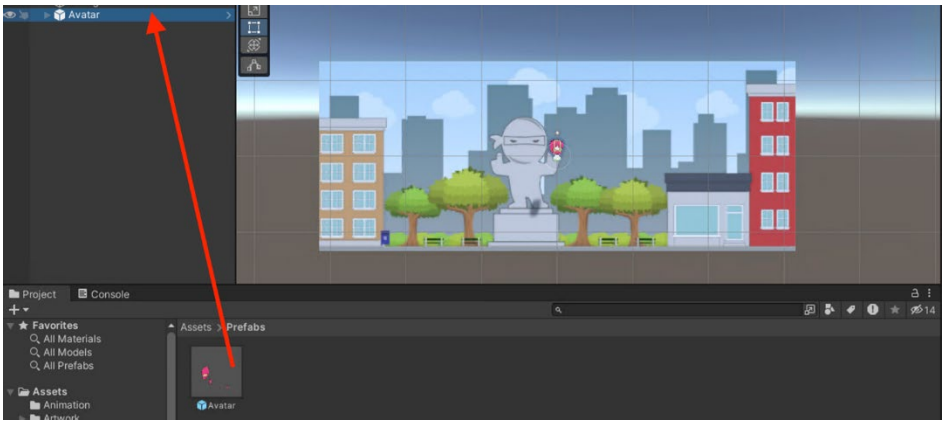
11 Now that there is a background image, you are ready to add a player character. Code Ninjas has a custom package put together for that. Click the **Assets** tab, select **Import Package** then **Custom Package**.

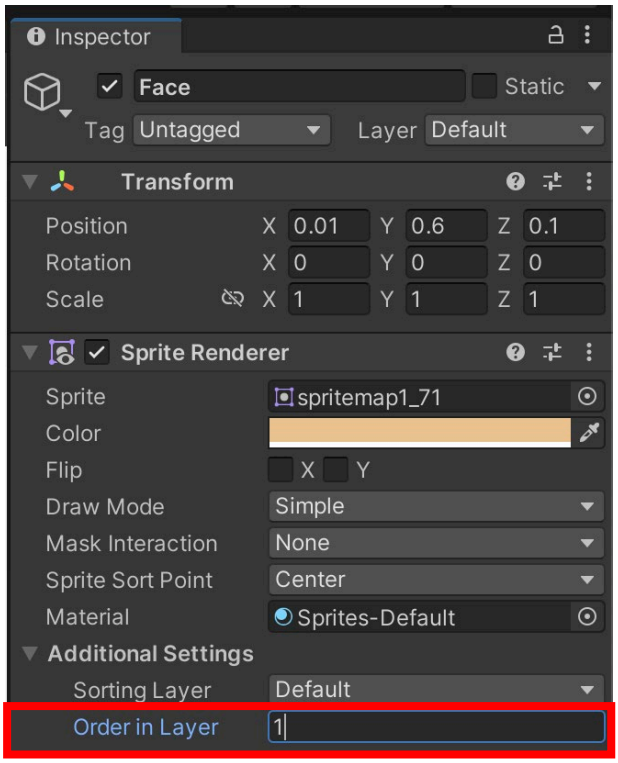


12 Navigate to the folder where your Bronze Belt assets are located and select **Activity 02 - CodeNinjasAvatar.unitypackage** to import the new player assets.

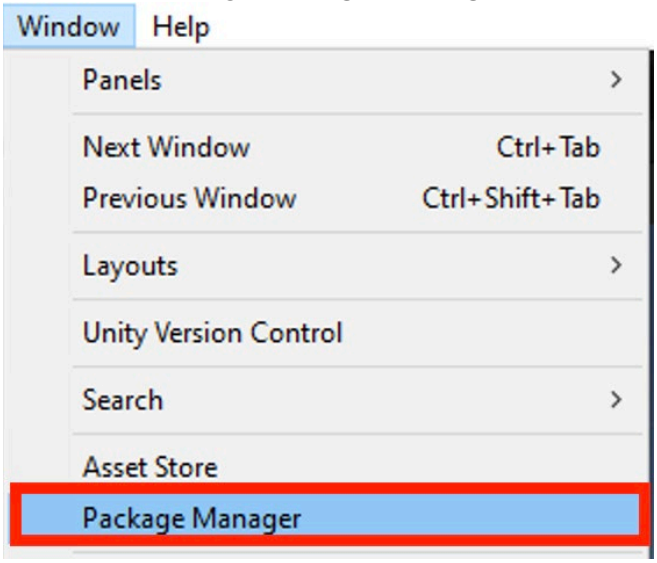


13 The package you just imported includes a new folder called **Prefabs.** Open this folder to find your **Avatar**. To add the avatar to your scene, drag it from the **Projects** panel into the **Hierarchy** panel. If the avatar appears to be missing a body part, select the drop-down arrow to the left of the **Avatar** in the **Hierarchy**. Then, select the missing part and change the **Order In Layer** to a number equal to or greater than 1. In the screenshot below you can see an example of this done for the face since it was not properly appearing. **Prefabs** will be covered in a later activity so for now, know that it is a special **GameObject** with everything it needs.





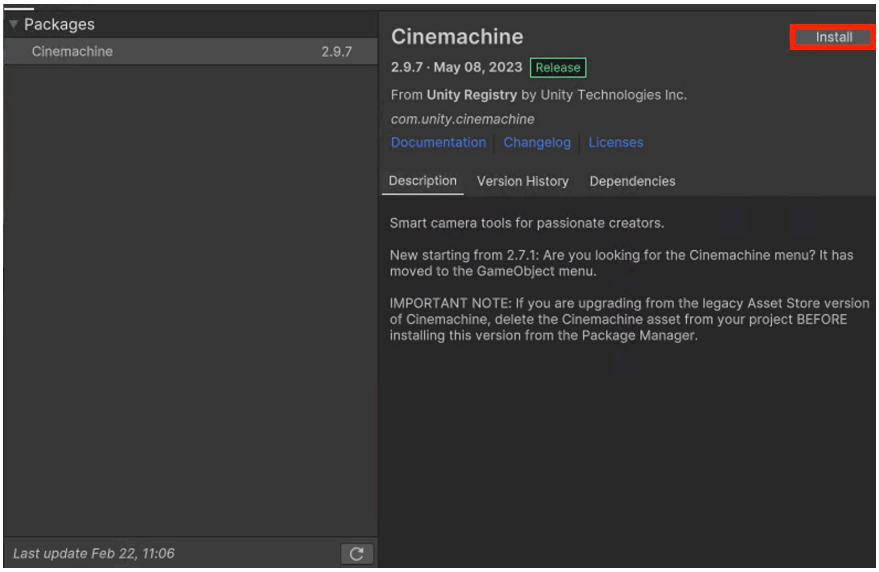
14 In addition to the custom packages, there is another type of Unity package that you can import. These can be found in the **Package Manager**. Open the **Package Manager** by clicking the **Window** tab and selecting **Package Manager**.



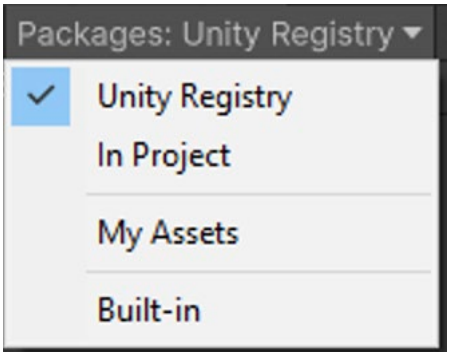
Pro Tip:

If the package that you are looking for doesn't appear immediately, it may be because Unity is still collecting all the information on available packages. Just give Unity a moment to load all this information.

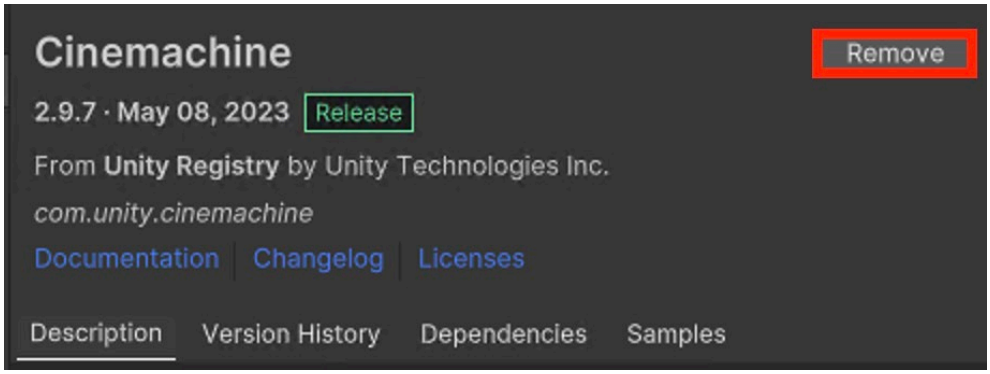
15 The package that you'll want to add is **Cinemachine**, a set of virtual camera tools that are ideal for the type of platforming game that you will be making. You may need to change the menu appearance from Packages: In Project to Packages: Unity Registry. In the search bar type **Cinemachine** then from the menu select it and click **Install**.



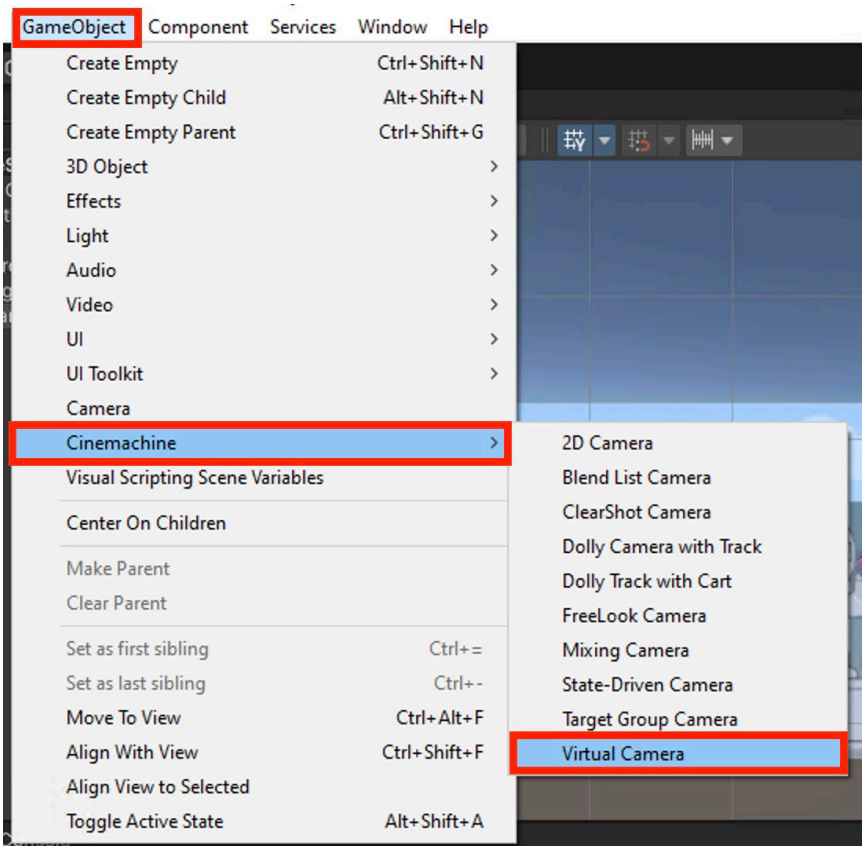
If you cannot see **Cinemachine**, make sure at the top it says **Packages: Unity Registry**, not *Packages: In Project*.



16 Wait until the button changes to "**Remove**" before continuing.

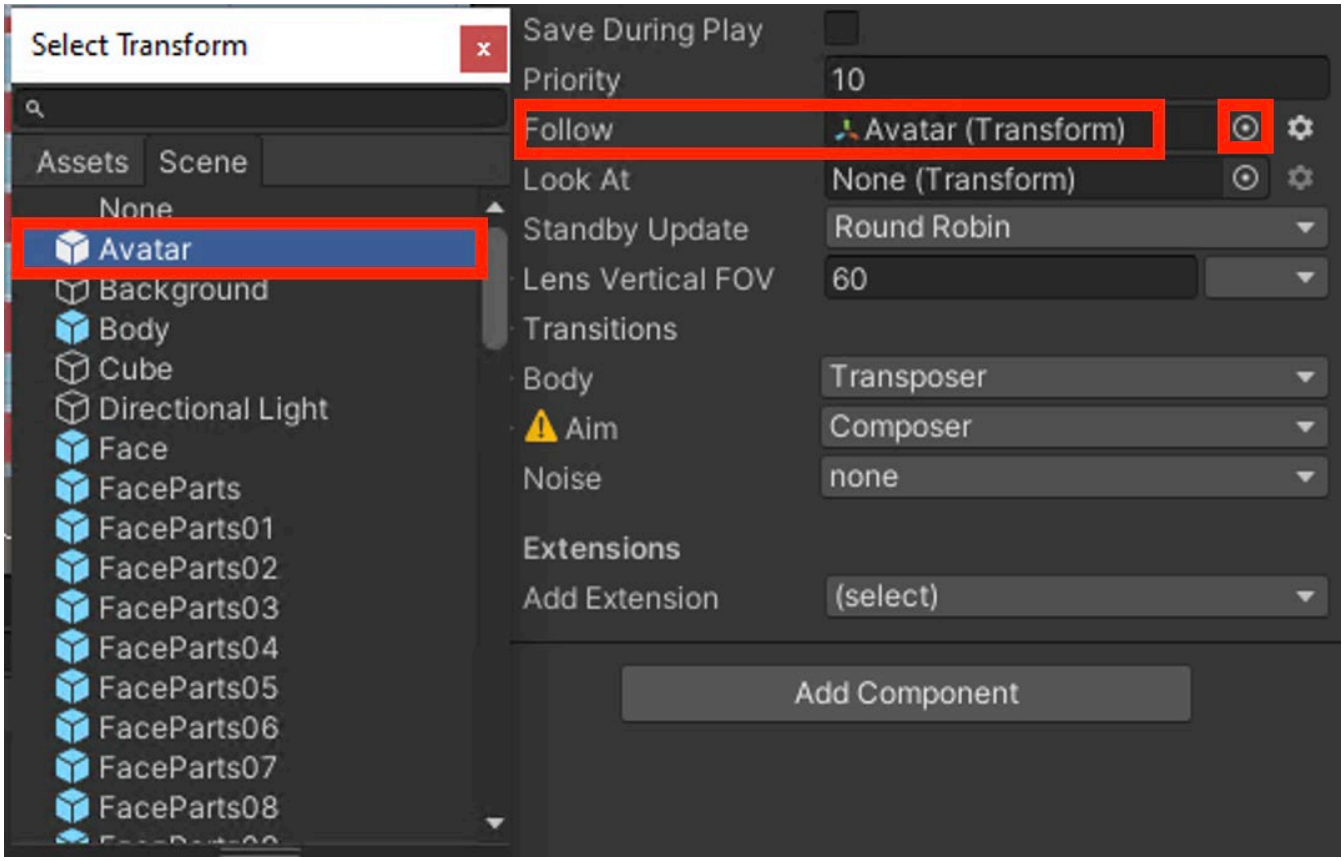


17 The package has added a new option in the **GameObject** and **Component** menus called **Cinemachine**. To find it, click on the **GameObject** menu and locate **Cinemachine**. Click on it to see many options for cameras in this menu. You'll want the option that says, "**Virtual Camera**". Click that option to add the **Cinemachine Virtual Camera** to your **Hierarchy**.



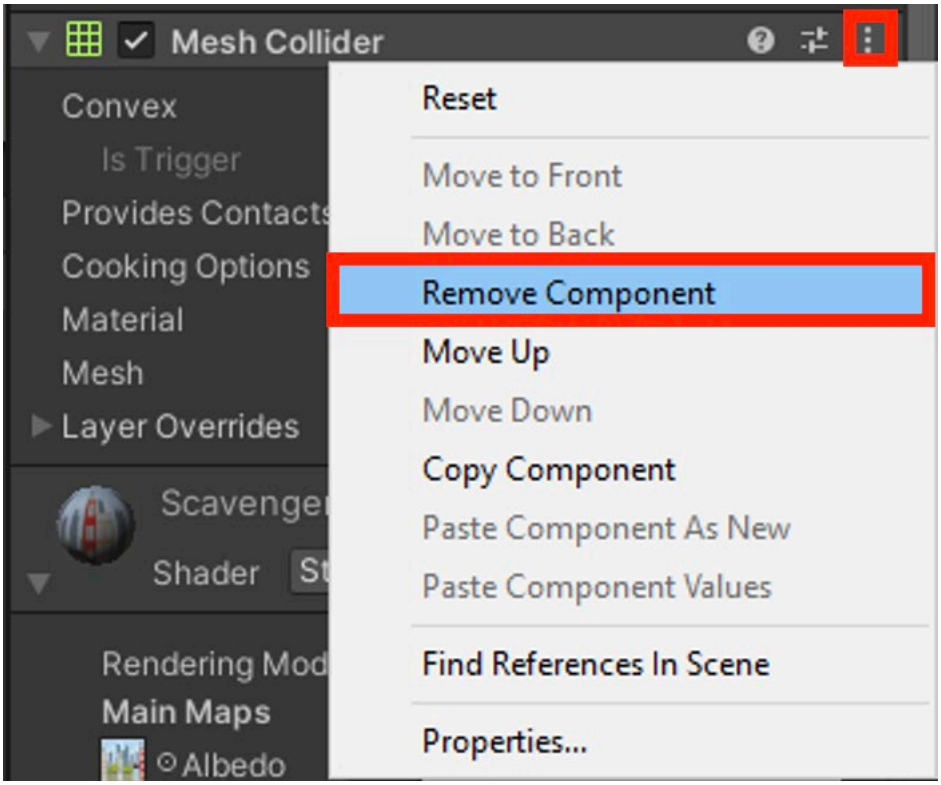
18 Click the **Virtual Camera GameObject** in the **Hierarchy** to set it up. In this game, we want the camera to follow the player wherever they go in the scene.

In the **Inspector**, find the **Follow** property and click on the small circle next to **None** to select a target for the camera to follow. This opens a menu of possible **GameObjects**. Click the **Avatar** to select it.

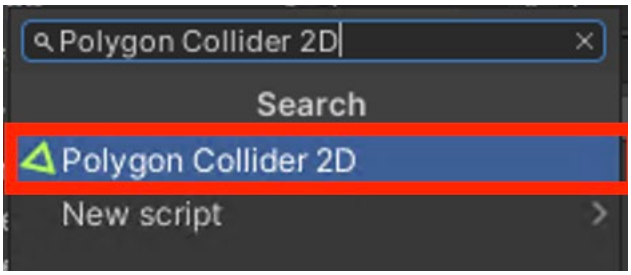


19 The **virtual camera** will now follow the player throughout the scene and beyond. However, the camera shouldn't go past the edges of the background. There needs to be some way to tell the **virtual camera** that the edges of the background are as far as it should go. To do this, you'll need to add a special collider to the background. But before that, we need to remove the current collider.

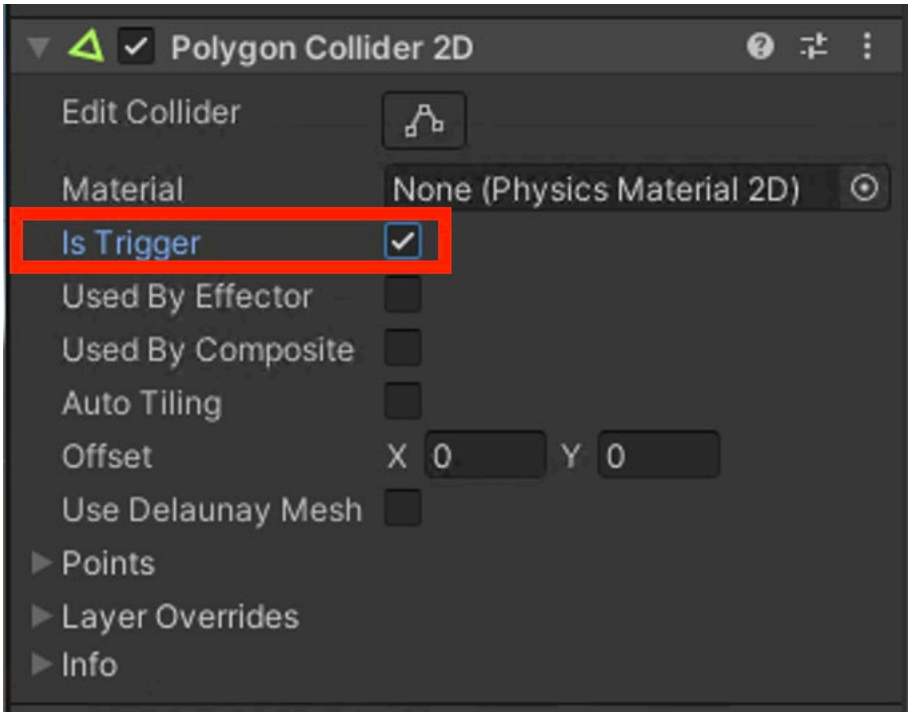
In the **Inspector**, select the **Background** object and find the **Mesh Collider** component. Click the three dots drop down menu in the right corner of the component to open the menu. Click **Remove Component** to remove it from the **GameObject**.



20 To do this we need to add a colider, click **Add Component** at the bottom of the **Inspector**. In the search bar type **Polygon Collider 2D** and then select it.

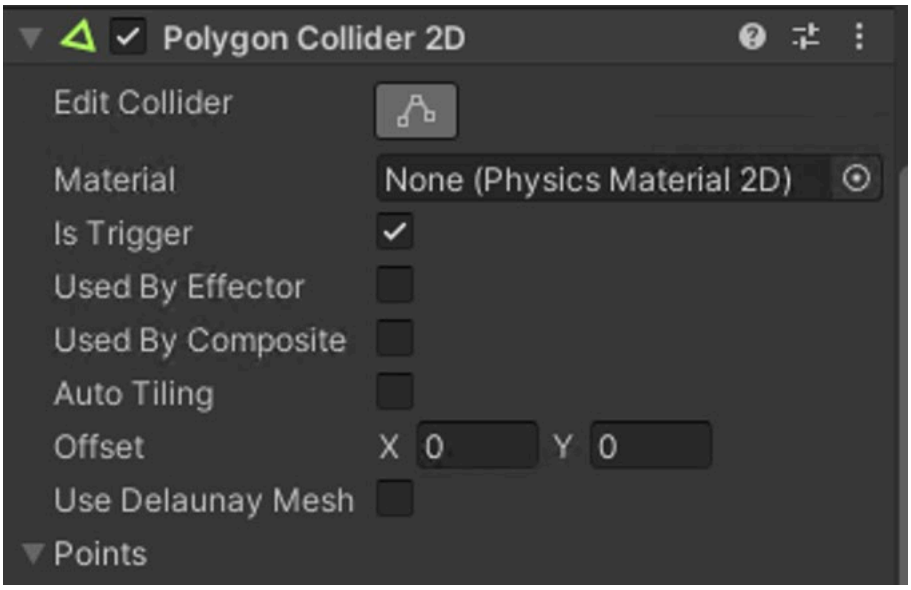


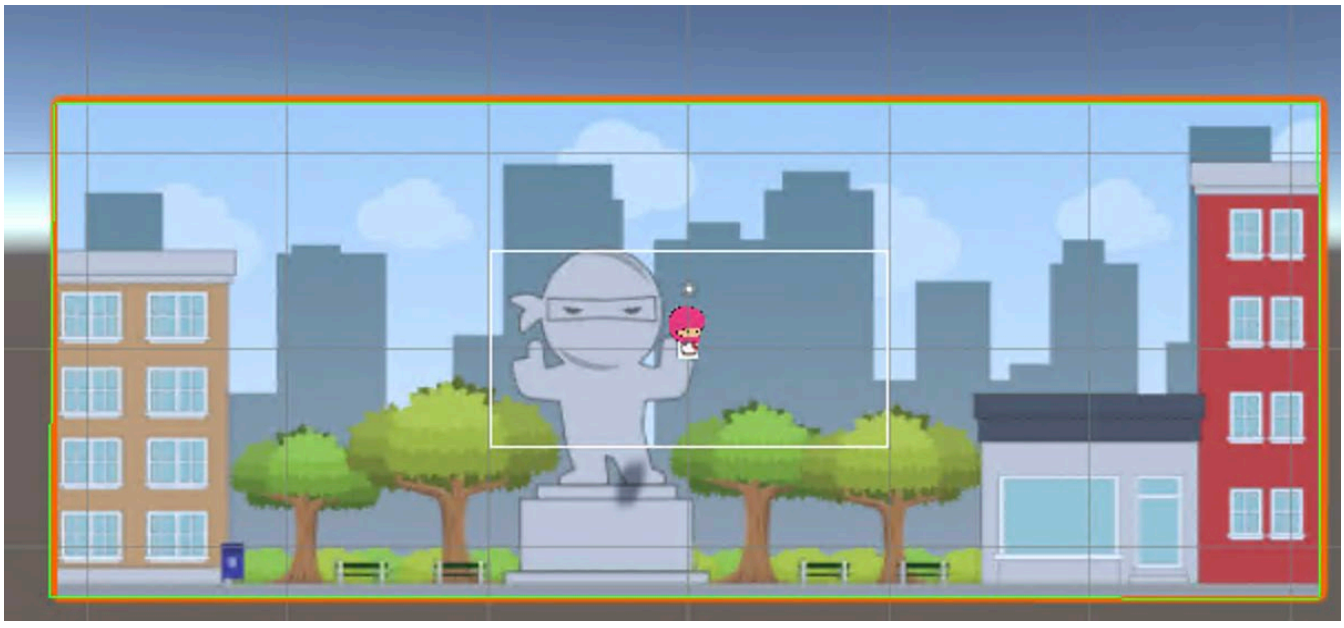
21 You should now see the **Polygon Collider 2D** component in the inspector window. Make sure that **Is Trigger** is checked.



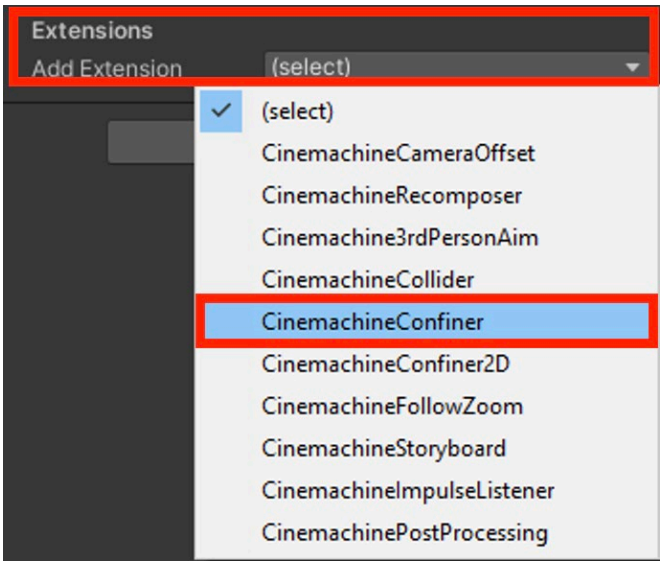
IMPORTANT: If you do not check **Is Trigger**, Unity will treat the entire background as a solid object and move the player outside of it!

22 The **Polygon Collider** needs to be adjusted to match the background. Click **Edit Collider** and see if the collider is already at the corners of the background. If not then drag the corners of the collider to the corners of the background. Any extra corner points can be removed by opening "**Points**" then "**Paths**" and then "**Elements**" in the component. Then, right click the point that you want to delete and remove it.

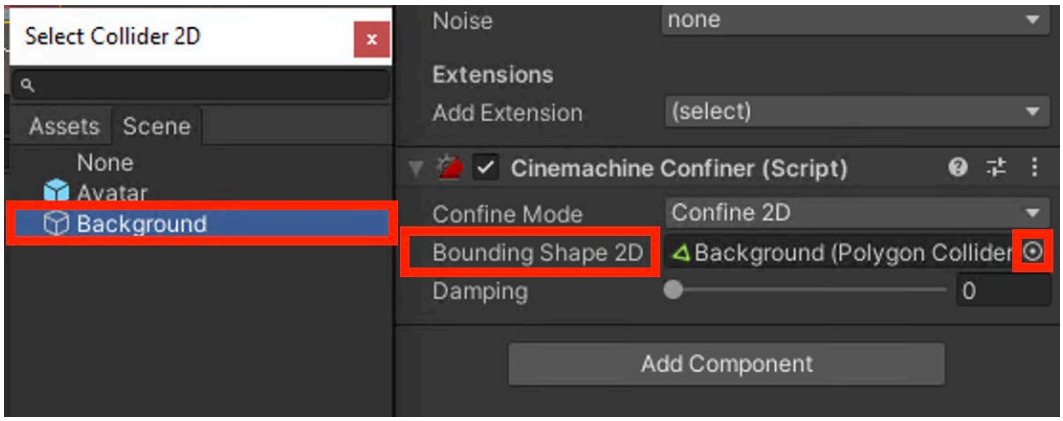




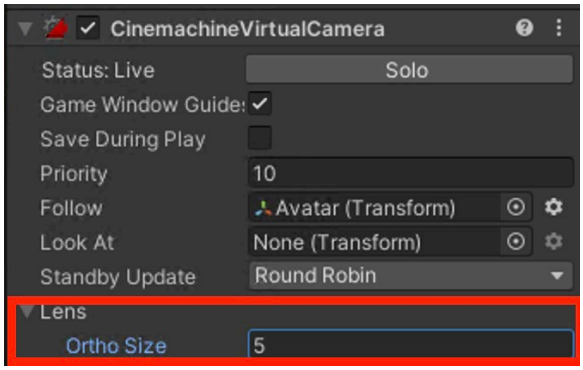
23 The final step of configuring the virtual camera involves specifying which **GameObject** contains the boundaries it should follow. To do this, locate and select the "**Virtual Camera**" **GameObject** in the **Hierarchy**. Then, in the **Inspector** panel, scroll down to the bottom and click "**Add Extension**." From the dropdown menu, choose "**CinemachineConfiner**."



24 **Cinemachine Confiner**, a new component, has been added to the Inspector. Where it says **Bounding Shape 2D**, we need to add the **Background GameObject**. This can be done either by clicking on the small circle and selecting **Background** from the menu that appears, or by dragging the **Background GameObject** from the Hierarchy panel into the slot for **Bounding Shape**.

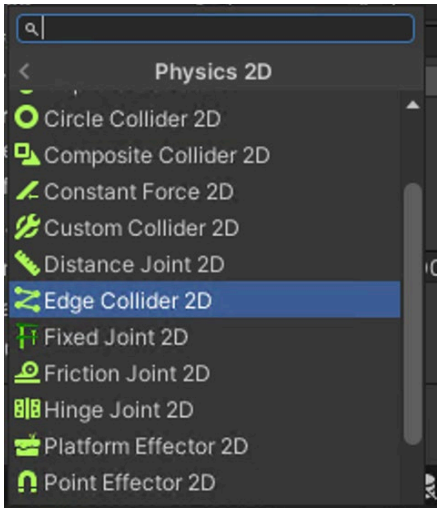


25 Now, let's adjust the **virtual camera** to bring it closer to the **Avatar**. In the **Inspector** for **Virtual Camera**, find the setting for **Lens**. If necessary, you can expand it by clicking the triangle to the left. Change **Ortho Size** to 5 to make the camera view closer to the **Avatar**.

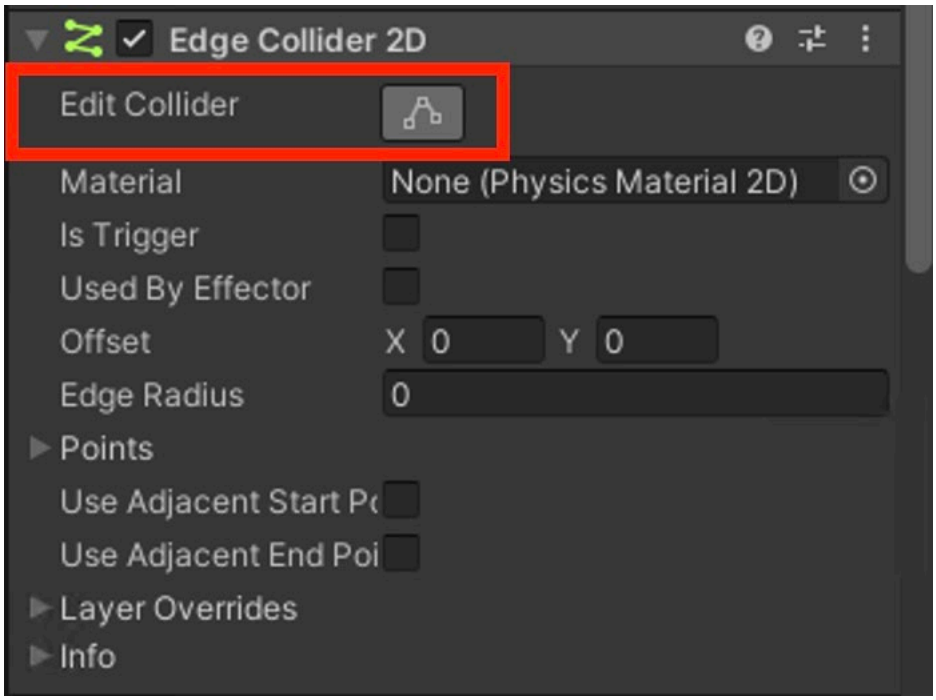


26 Test the camera by clicking the Play arrow above the scene. You'll notice that the virtual camera will follow the player until it falls off the edge of the screen. Since we've confined the camera, it won't move past the edge of the background. Unfortunately, there's nothing stopping the Avatar. Let's fix that. Click the Play button again to stop the game.

27 To keep the player within the scene, we can use a collider. If you added an ordinary **Box Collider**, Unity would try to put the player outside of the box. Fortunately, there's a collider for situations like this called the **Edge Collider**. In the **Inspector** panel for the **Background**, click **Add Component**, select the **Physics 2D** menu then select **Edge Collider 2D**.

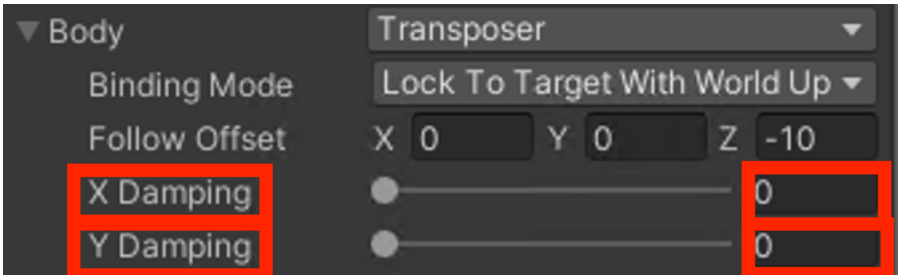


28 The **Edge Collider** might be hard to see at first since it's nothing more than a straight line. Click **Edit Collider** to adjust it to match the background.



29 Editing the **Edge Collider** follows the same process as editing the **Polygon Collider** which you have now done. You can simply click and drag one end of the collider over to where you want it to be. You can add more points and make corners by holding the cursor over the middle of the line. A draggable point will appear there. Stop once you have made a closed box around the edge of the background.

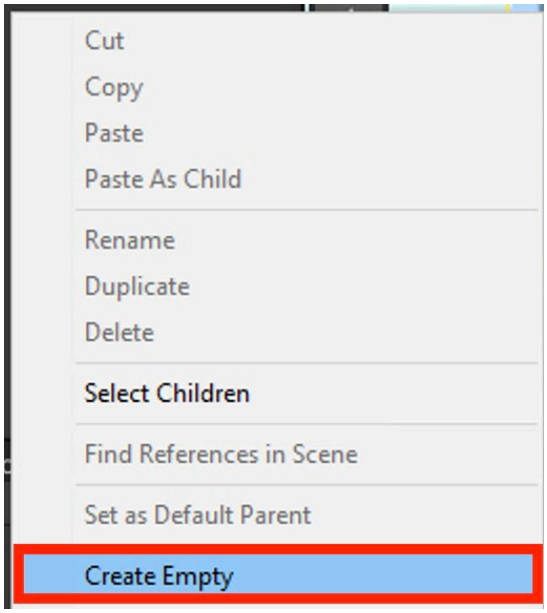
In the **Hierarchy** select the **Virtual Camera GameObject**. In the **Inspector** panel open the "**Body**" section using the triangle. Adjust the "**X Damping**" and the "**Y Damping**" to 0. This ensures the camera does not get jittery.



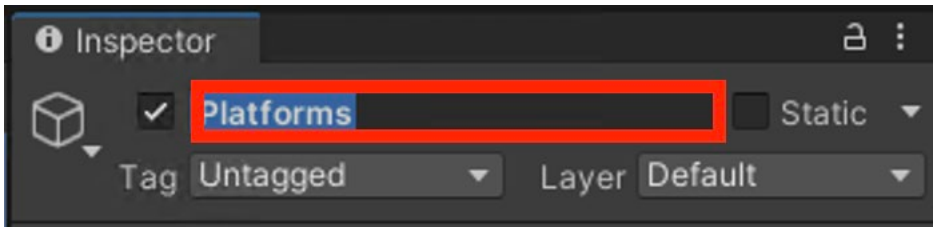
Pro Tip:

The edge collider is exactly where the Avatar stops moving, so feel free to place it above the sidewalk and inside the edges of the background to make the Avatar have a more natural look and feel when it moves. You can test this by playing the scene until everything looks good to you.

30 The next step is to create more colliders for the **Avatar** to walk on. To keep things organized, we'll create a place in the scene to hold all the platforms. In the **Hierarchy** panel, right click and select **Create Empty** to create an empty **GameObject**.



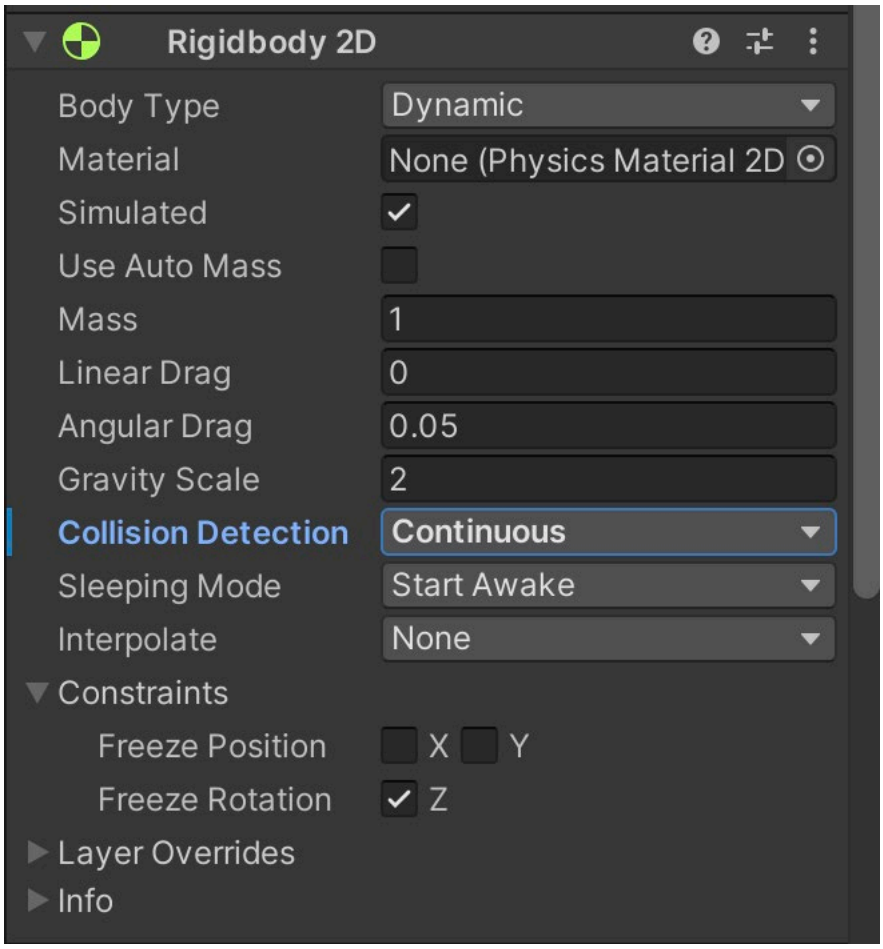
31 Change the name of the new **GameObject** to **Platforms**.



Pro Tip:

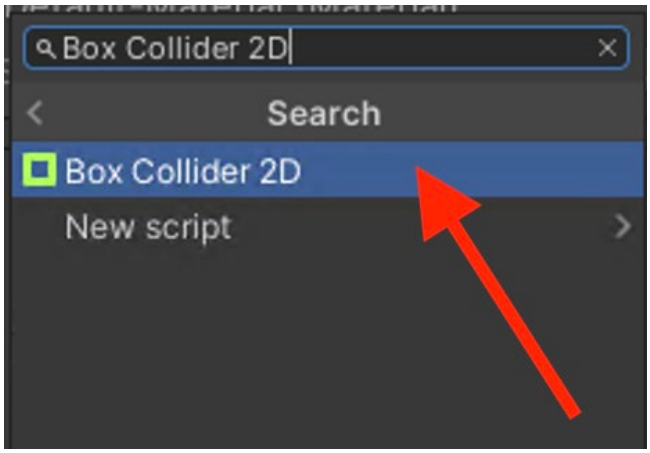
In the **Hierarchy** an empty **GameObject** can serve the same purpose as folders in the **Project** panel. In addition, since they are **GameObjects**, you can add **components** to them so that they can perform other functions in your game!

32 Before we begin creating Platforms select your **Avatar** in the **Hierarchy** and find your **Rigidbody2D**. Set the **Collision Detection** for your Rigidbody2D to **Continuous**. In addition check to ensure that the Z box is checked for Freeze Rotation.



33 To begin adding platforms, right-click on the **Platform GameObject** in the **Hierarchy** and select **2D Object** then **Sprites** and finally **Square** to add a Square to the scene inside the **Platform** object.

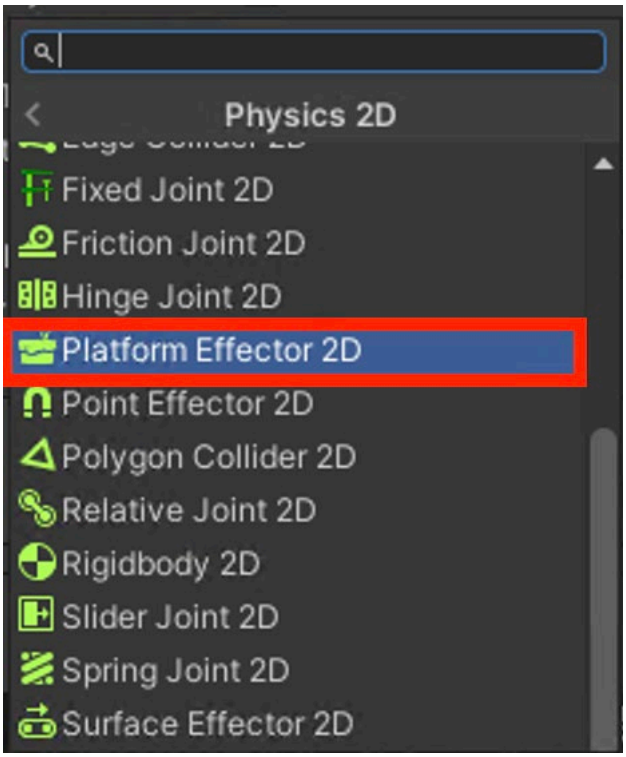
34 In the **Inspector** panel, select **Add Component**, then add **Box Collider 2D** to the square. Adjust the size of the square so that it matches the shape of the window ledge.



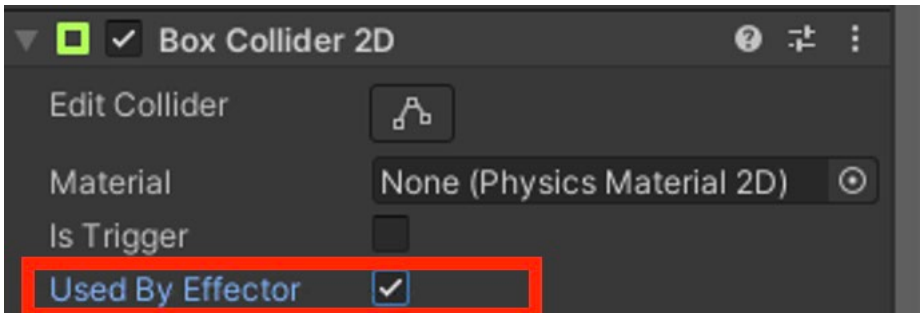
Pro Tip:

Don't forget, you can easily move around in the scene panel by using the mouse. Use the mouse wheel to zoom in or out of the scene and hold down the right mouse button and drag to get to the part of the scene that you need to work with.

35 Currently, the **Square** will block the player from all directions. However, what we truly want is a way for the collider to stop the player only when they land on the collider from above, and to let the player pass through it from all other directions. To achieve this, you'll need to add another component to modify the **Box Collider** that was just added. Click **Add Component** again and select **Platform Effector 2D**.



36 Next, we need to configure the two components to work together. With the **Square** still selected, go down to the **Box Collider 2D** component and make sure that "**Used By Effector**" is checked. This means the **Platform Effector** settings will modify the **Box Collider 2D** component. In the **Platform Effector** component, keep the default settings as they are.



37 Let's test our platform. Begin the game by clicking the Play button. Use the arrow keys or WASD to move the avatar over to the ledge and then use the space key to jump onto the ledge. Now that there is a working platform object, you can simply copy it over to where it's needed.

Stop the game by clicking the **Play** button again.

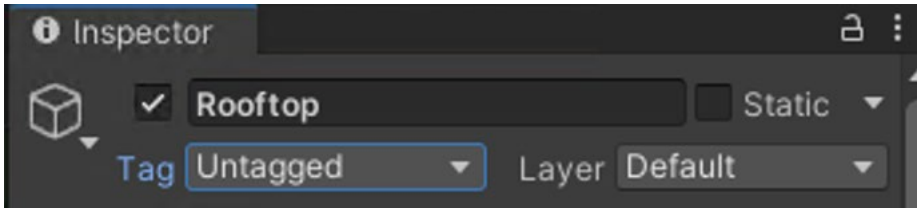
Pro Tip:

Don't worry if the platform object is still visible in front of the background. That will be fixed in the next step.

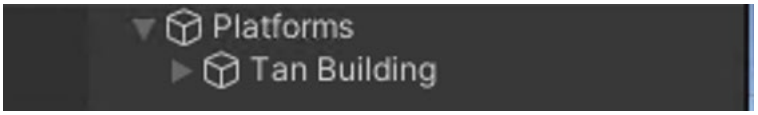
38 Before proceeding, adjust the position of the **Square** so that it is positioned behind the background by changing the **Order In Layer** value to **-1**.

With the **Square** still selected, rename it to **WindowLedge**. Then, use **Ctrl+D** to duplicate **WindowLedge** and move the duplicate over to one of the other windows. Select both the original and the copy and press **Ctrl+D** again to copy both objects and move the copies over to new windows. Select all four platforms and press **Ctrl+D** again to create enough window ledges for the entire building.

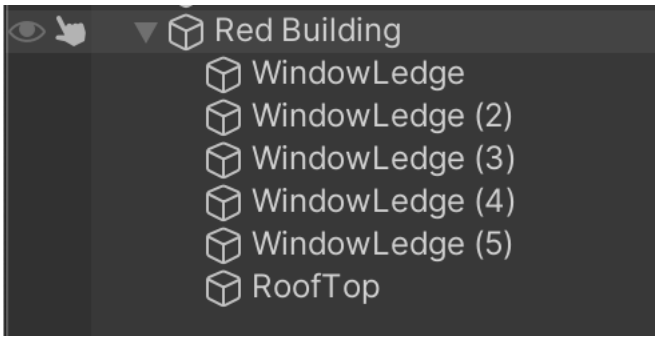
To complete the building, make a single copy of a ledge and move it to the roof. Rename the platform **RoofTop** and adjust its size to cover the length of the roof.



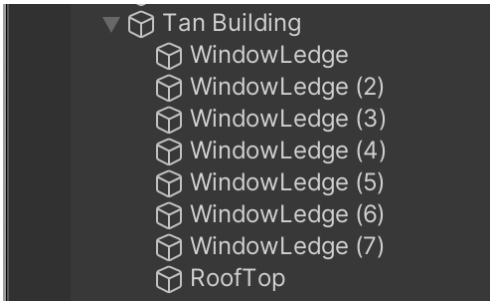
39 To keep things organized, right click on the **Platform GameObject** in the **Hierarchy** panel and create an **Empty Object**. Rename this object ***Tan Building*** and move all the objects you just made so that they are now part of the **Tan Building GameObject**.



40 While the **Tan Building GameObject** is selected, press **Ctrl+D** to copy the object along with all the platforms inside it. Change the new **GameObject** name from **Tan Building** to ***Red Building*** and use the **Multiple Objects Tool** to move these new platforms to the red building on the right side.

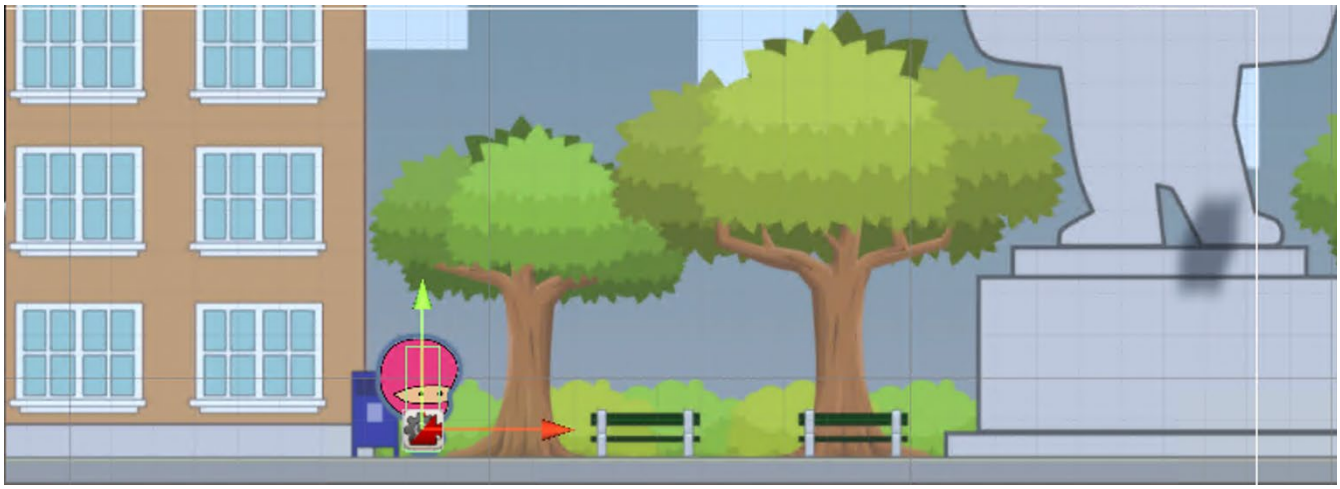


41 Similarly to how you handled the Tan Building, move the platforms for the window ledges and roof top to their proper places for the Red Building. Delete any platforms you don't need.



42 Continue with duplicating and repositioning the groups of platforms. Use this guide as a suggestion for grouping and placing the platforms.

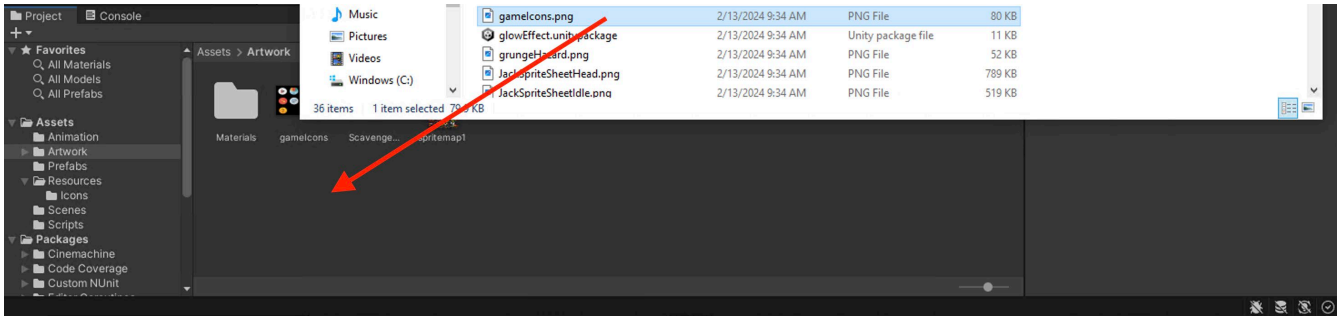
43 If it is not done already, position the Avatar on the ground. Here, we have placed it next to the mailbox.



44 Test the game by clicking the Play arrow. Make sure that your avatar can reach all parts of the scene that have platforms.

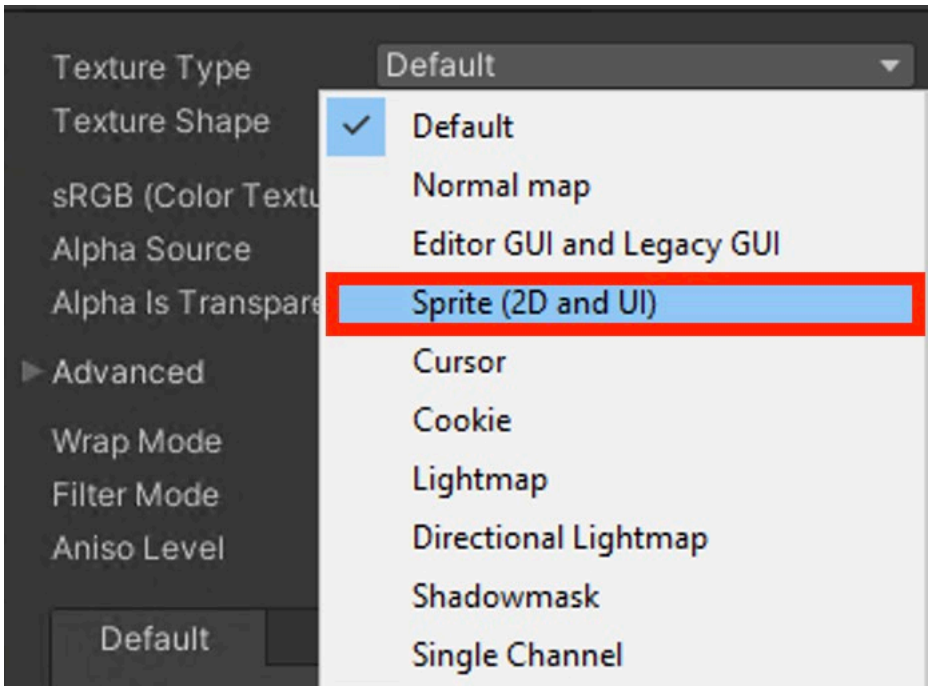
Stop the game by clicking the Play arrow again.

45 Let's provide our player with something to collect. In the **Projects** panel, make sure that the **Artwork** folder is open. Open File Explorer on your computer and navigate to where the Bronze Belt files are located. Find the file called **Activity 02 - gamelcons.png**. Click and drag it into the **Artwork** folder.

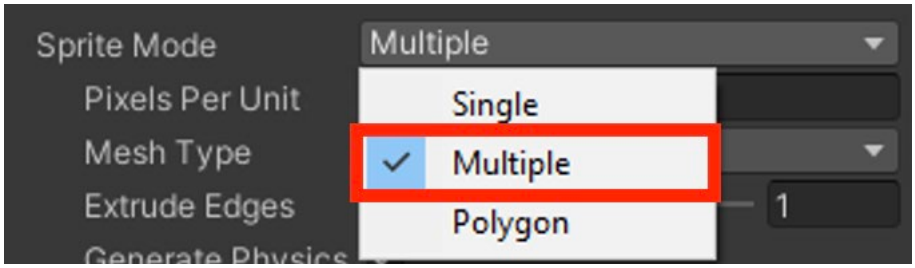


46 With the gameIcons selected in the **Projects** panel, navigate to the **Inspector,** and make two changes as follows:

Ensure the **Texture Type** is set to **Sprite (2D and UI)**.

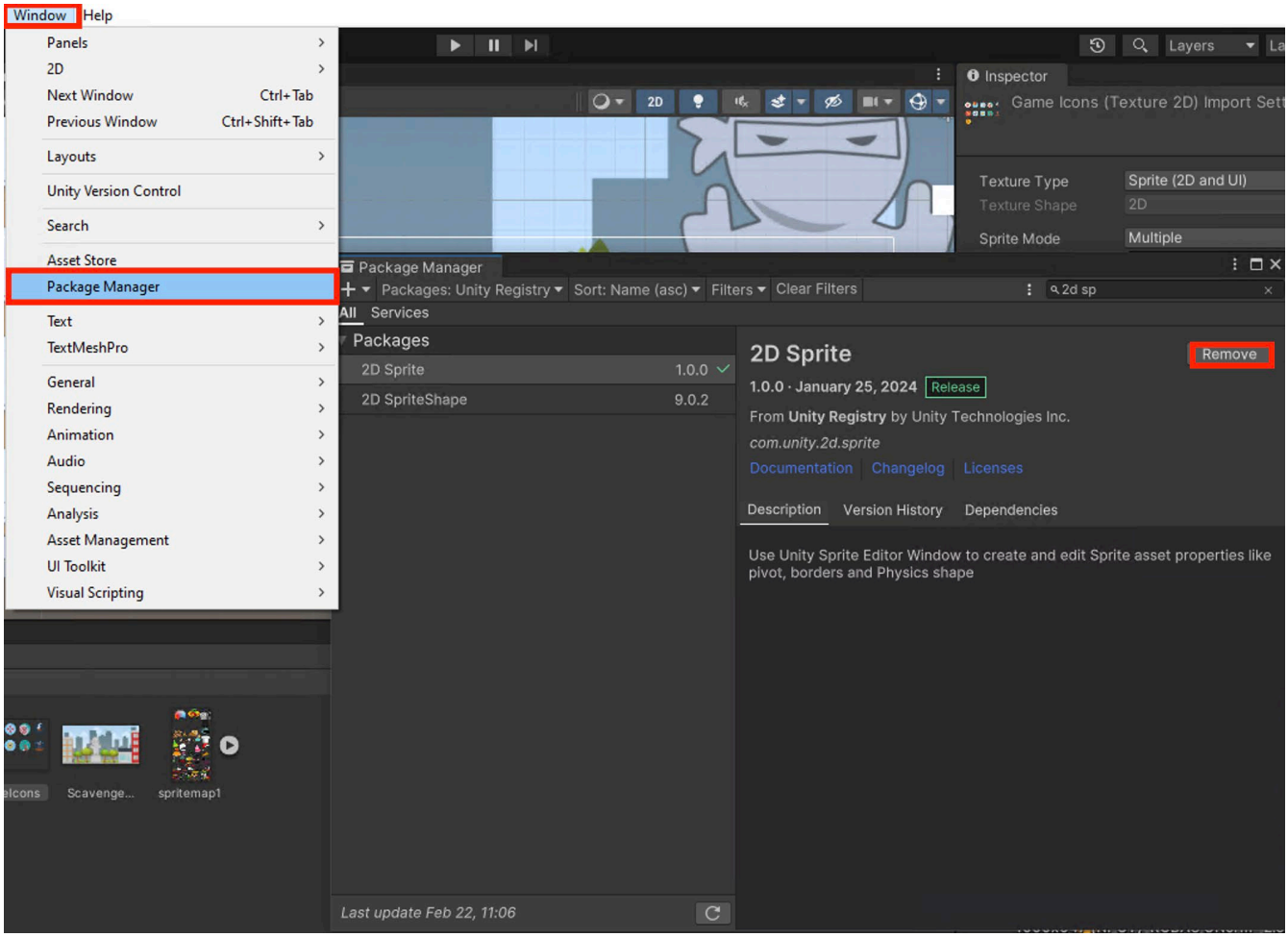


Set the Sprite Mode to **Multiple**.

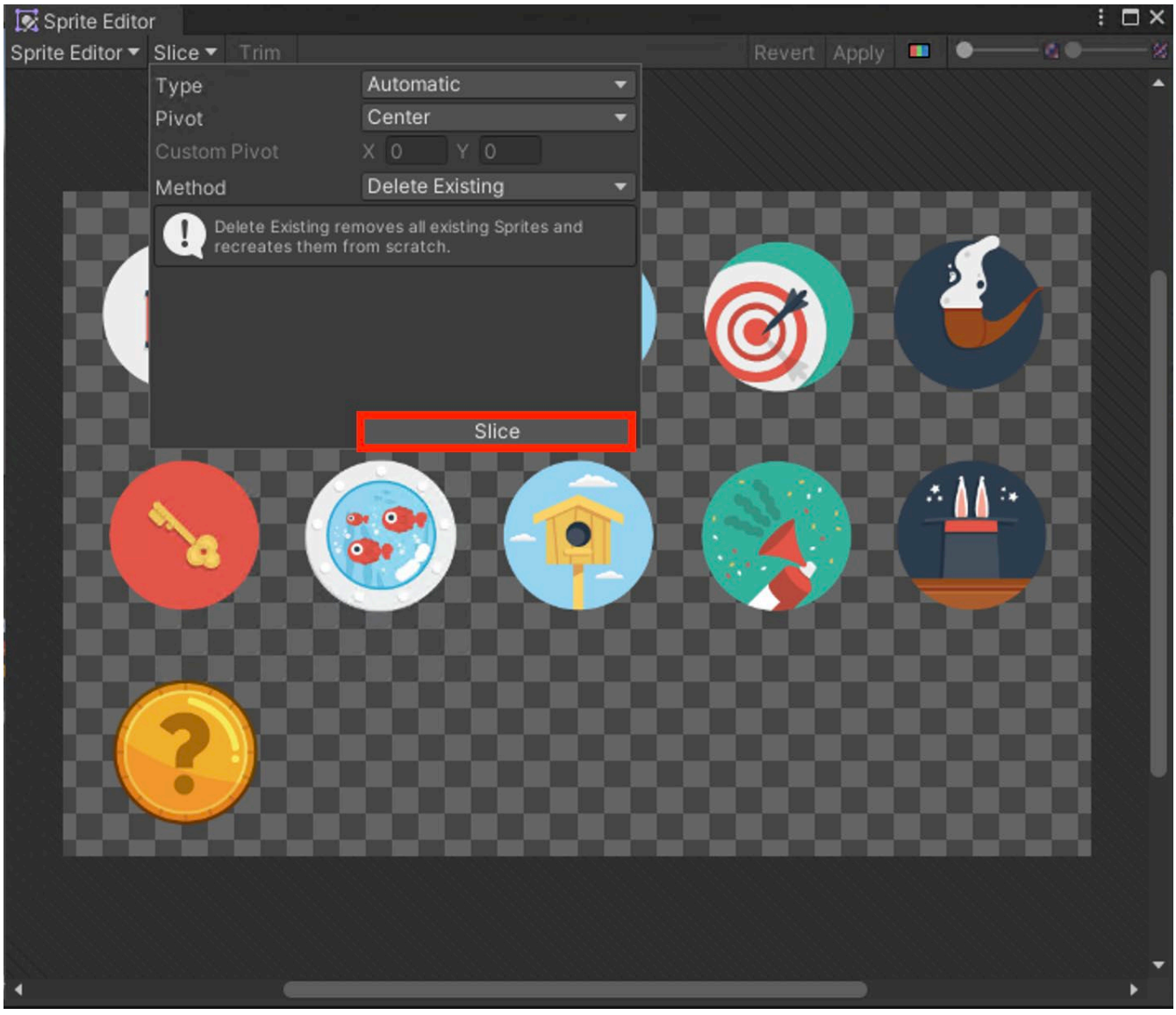


To save these changes click the apply button near the bottom right corner.

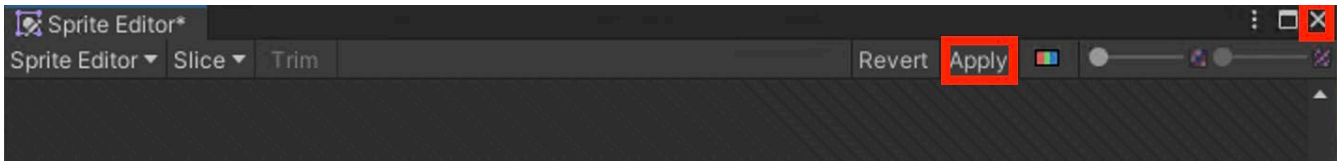
47 While you should have the **2D Sprite** package installed. Let's check by clicking the "**Windows**" tab in Unity and accessing the **Package Manager**. Locate and select the "**2D Sprite**" package. Click "**Install**" if prompted and wait for the installation to be completed before closing the Package Manager and proceeding. If it says "**Remove**" then you can exit Package Manager.



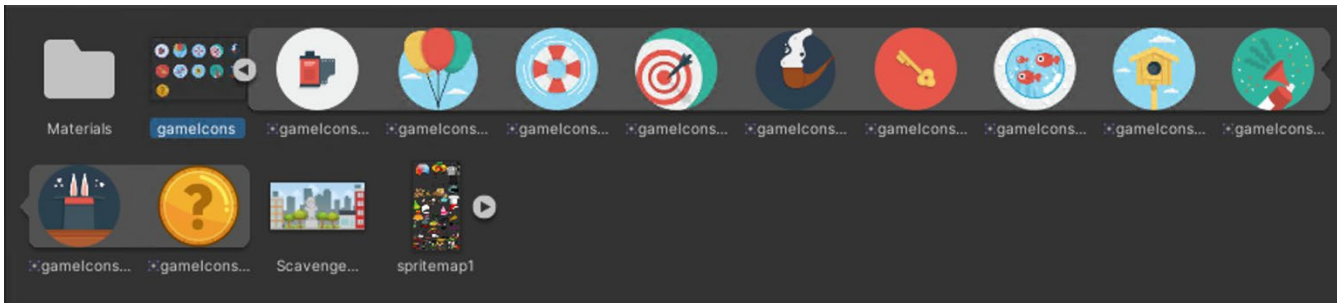
48 While still in the **Inspector**, click on the **Sprite Editor** button. By changing the mode to **Multiple**, we can now get many sprites from this single image. To do so, click on the **Slice** tab at the top of the **Sprite Editor**. For now, we'll keep the settings as is. Click on **Slice**.



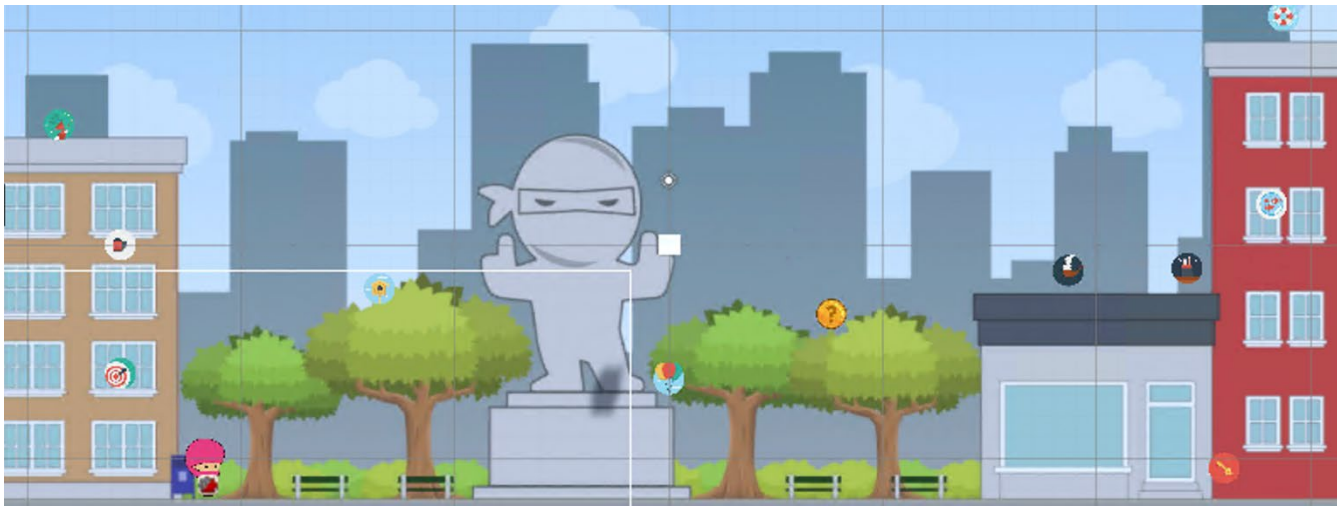
49 Now, all the icons on the sheet are defined as individual sprites. Click on any of them to see more details. Once you're done, click **Apply** and then close the **Sprite Editor**.



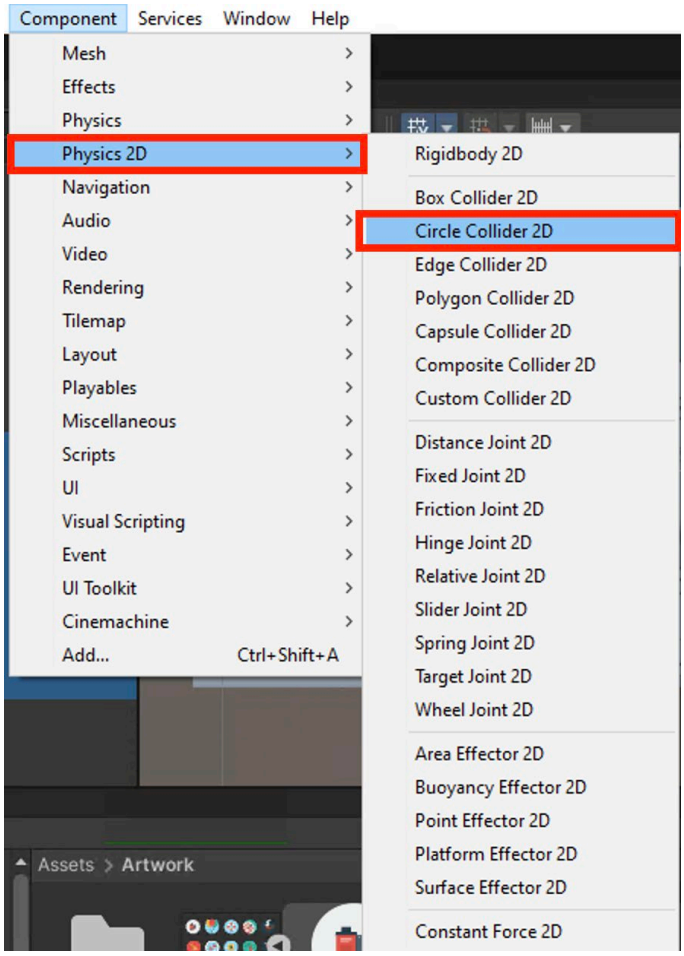
50 In the **Project** panel, the **gamelcon** asset now has an arrow on the right side. Click on it to see all the sprites.

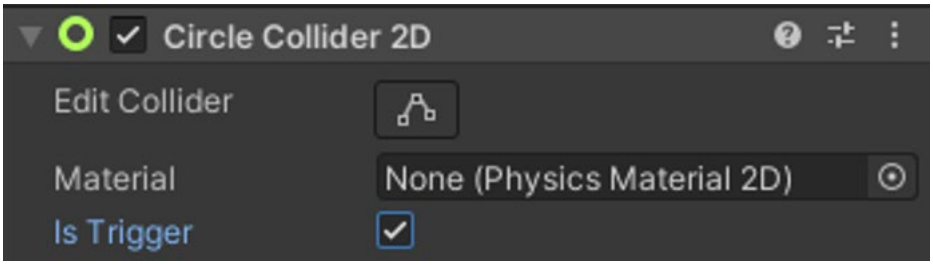


51 In the **Hierarchy** panel, create a new empty object and name it ***Collectibles***. Drag each of the new sprites into the **Collectibles** object within the **Hierarchy**, then position them throughout your scene in places where they might be challenging to reach.

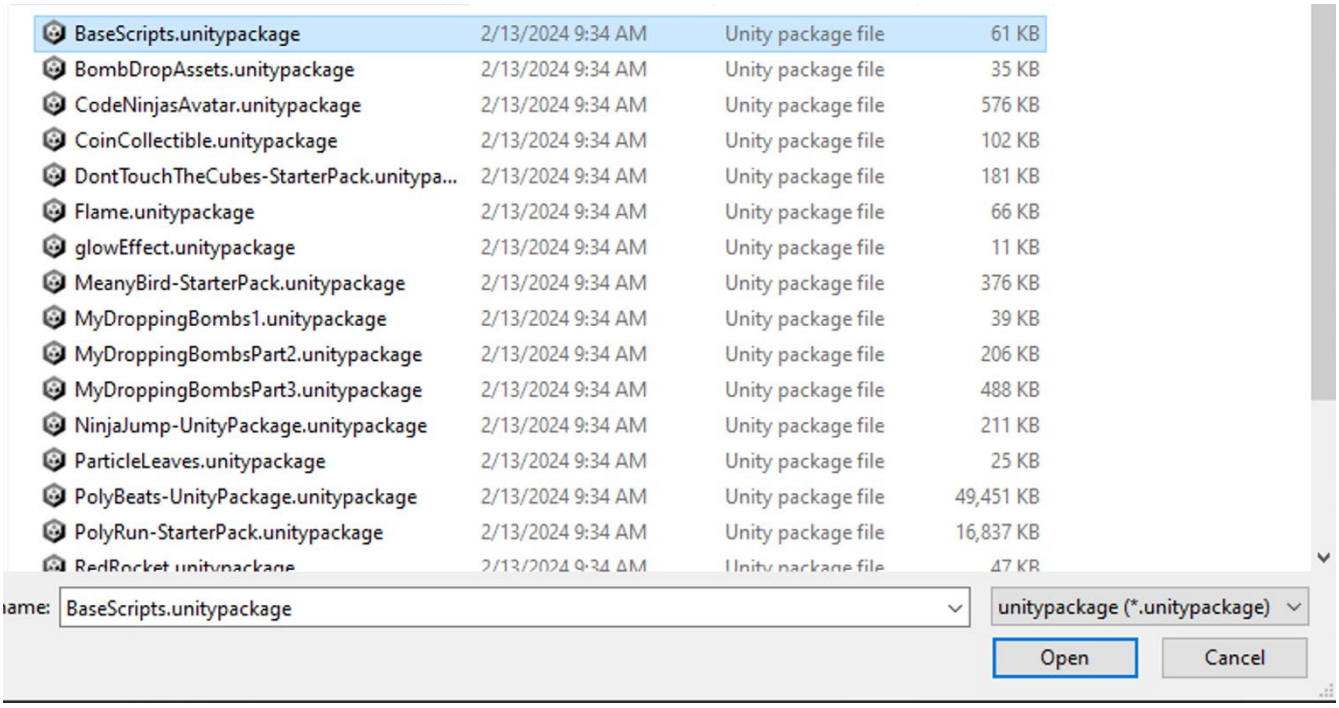


52 Currently, the sprites won't detect when the player touches them because they have no collider. Select all sprites in the **Collectibles** group and click the **Component** menu to add a **Circle Collider 2D** to them as shown. While all sprites are selected, ensure that **IsTrigger** is checked in the **Circle Collider** component.

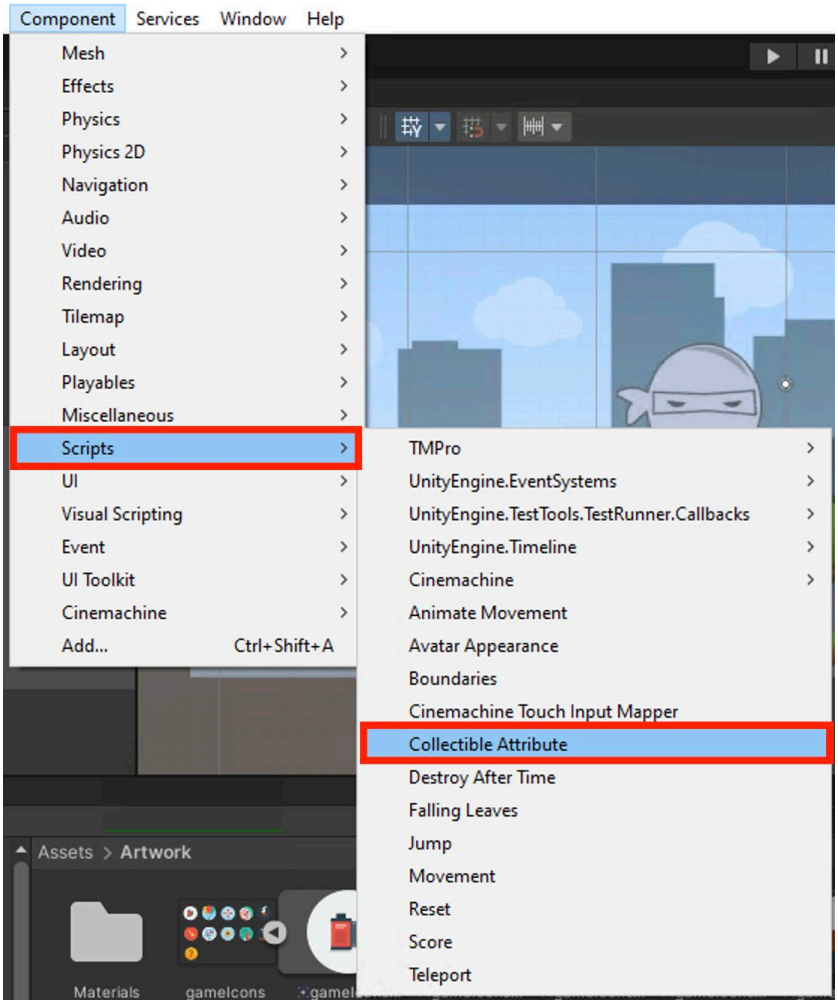




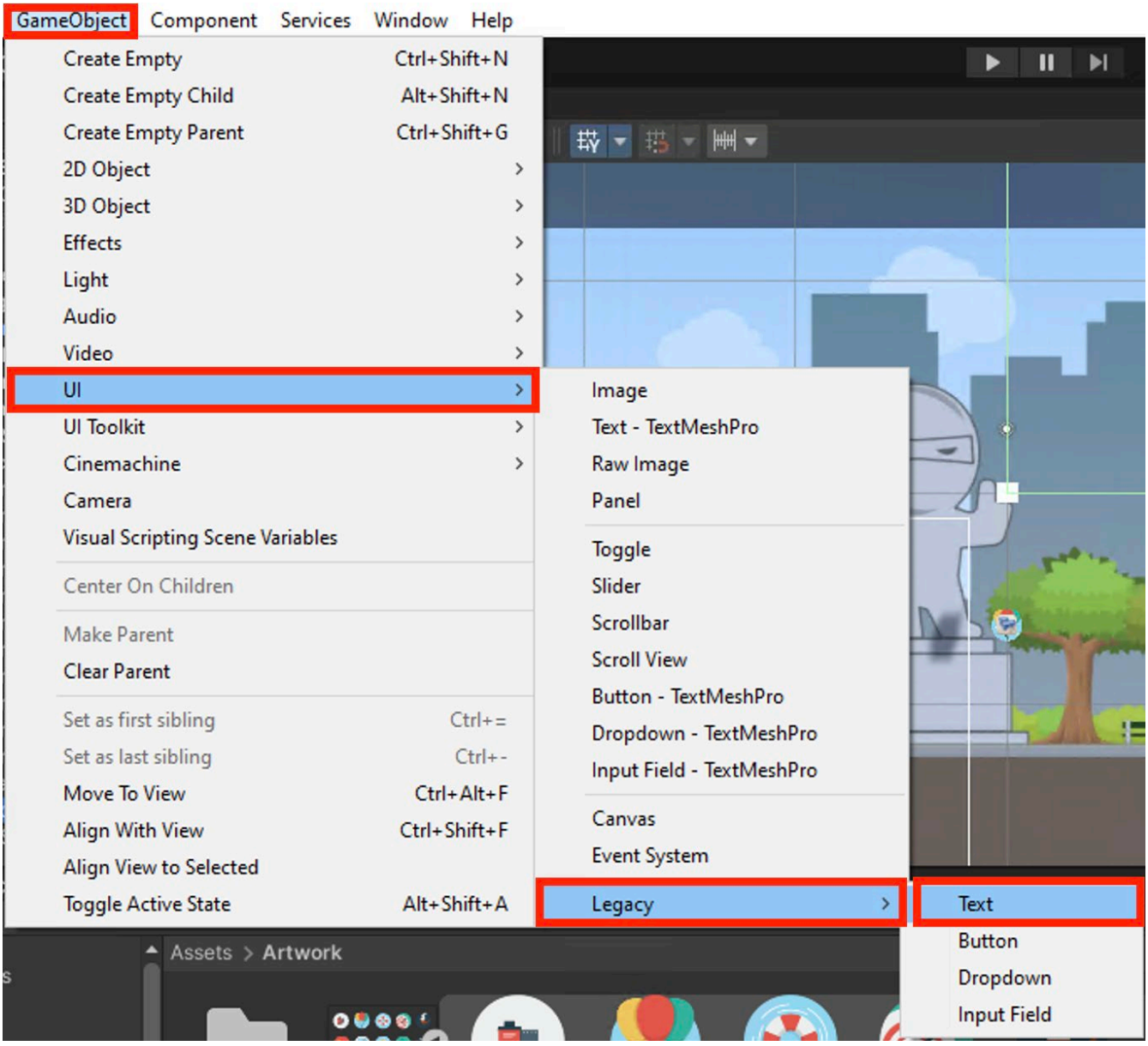
53 We also need a script to tell the sprites what to do. Click the **Assets** tab, then select **Import Package** followed by **Custom Package**. Navigate to your Bronze Belt files, then select **Activity 02 - BaseScripts.unitypackage** and import it.



54 You should now have some new scripts for the game. Highlight the sprites in the **Collectibles** object again, then click on the **Components** tab. Navigate to **Scripts**, then select the **Collectible Attribute** script to add it to all sprites. The script will have the sprites add points to your score when you touch them. But wait, you don't have a score just yet!

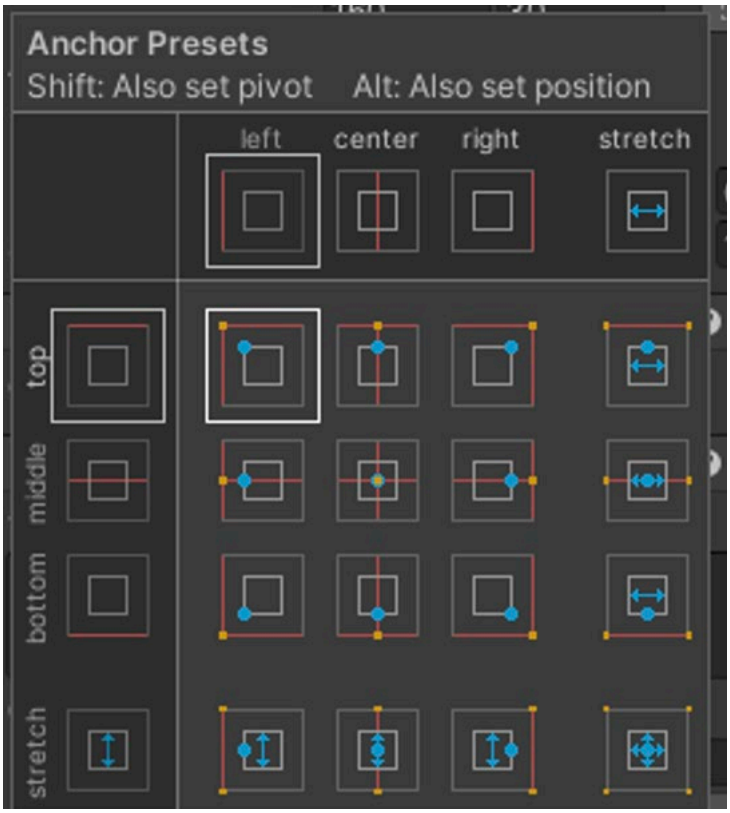


55 To display the score, you'll need a new **GameObject** called a **User Interface (UI)**. Click the **GameObject** menu, select **UI**, then **Legacy**, then **Text**.



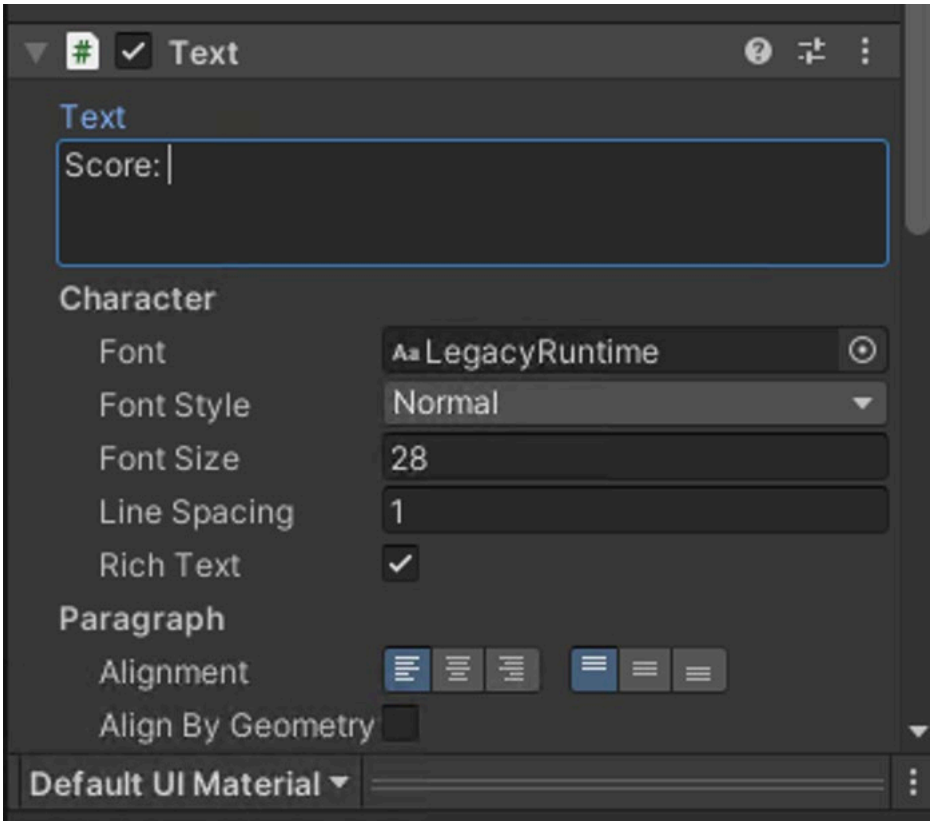
56 The **UI GameObject** automatically generates an object named **Canvas** to hold all **UI** elements. However, you probably won't see anything right away.

Next to the **Scene** tab, there is a **Game** tab that allows you to preview the game screen. Select the **Text** object in the **Hierarchy** and click on the **Rect Transform** square in the Inspector to expand it. This provides options for adjusting where the text appears. While holding down the **Alt** key, click on the upper left square in the grid to move the text to that corner. Now the text should be visible, but it may appear small.

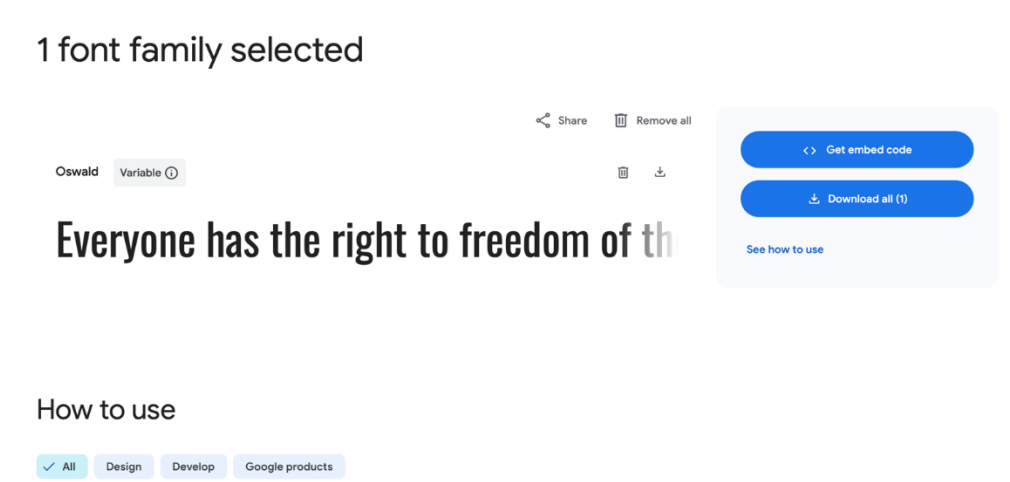


57 Click back to the **Scene** tab. With **Text** still selected, press the **F** key or double-click to focus on the **Text** object. If pressing **F** doesn't work, try **Shift + F** to focus.

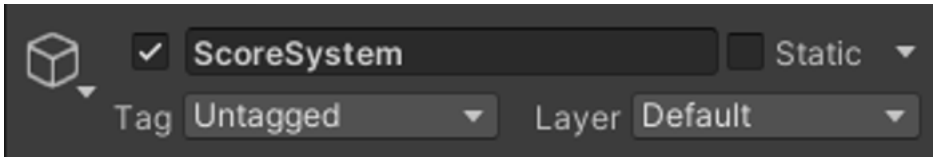
Now, let's fix the size of the **canvas**. Unity ensures that the **UI** appears correctly in complex 3D environments by defaulting to a large canvas size. In the **Inspector**, scroll down to the **Text** component. Change the **Text** to "***Score:*** " (don't forget the space after the colon!) and increase the font size to **28**. Even though the font is larger, the Text object stays the same size. Use the **Rect Tool** to make the object large enough for both your font and your score. Also, move the object so that it isn't touching the edges of the **Canvas**.



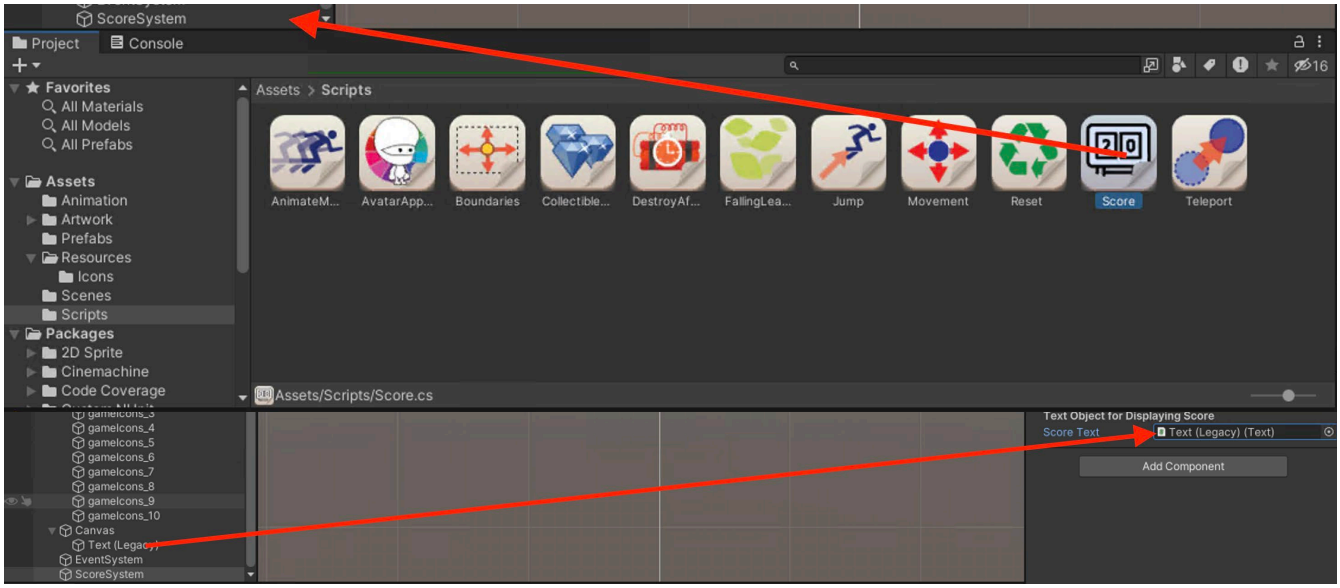
58 If you want to, you can change the font. There are plenty of fonts available for download at *fonts.google.com*. If you find one you like, select it and download it to your computer. Google fonts are in a .zip file that needs to be unpacked, but then you can add the fonts to your game just like any other asset.



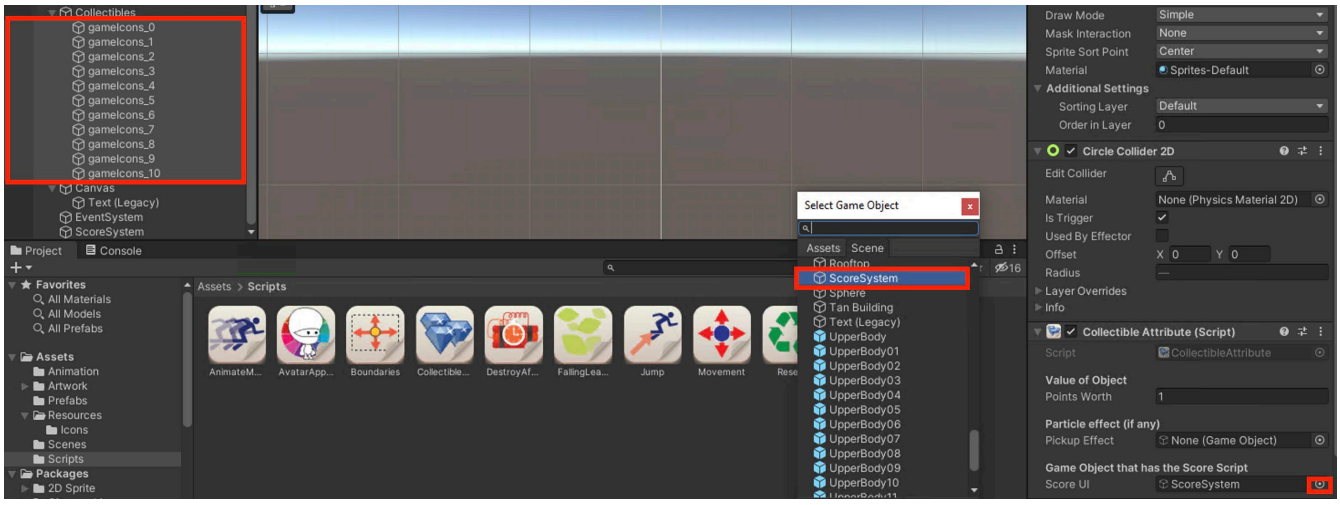
59 Let's return to our scoring system. In the **Hierarchy** panel, create a new **Empty Object** and name it ***ScoreSystem***.



60 Open the **Scripts** folder and drag the **Score** script into the **ScoreSystem** object. In the script component, there's a property for the **Text** object. Drag the **Text** object from the **Hierarchy** panel into the **Score Text** field in the script component.



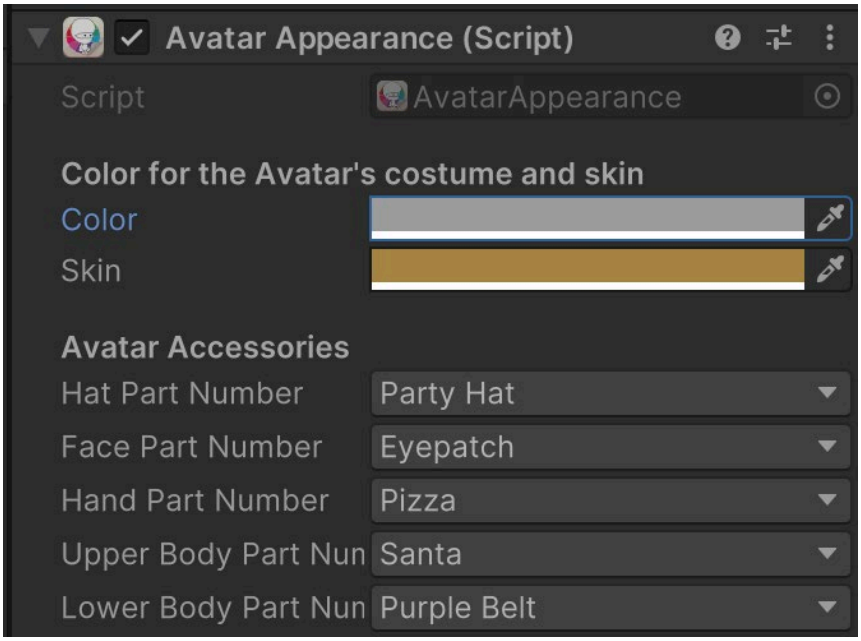
61 To connect the **Collectibles** to the **ScoreSystem**, select all the sprites within the **Collectibles** object. In the **Inspector** panel, find the **Collectible Attribute** script. There is a slot for the **Game Object with the Score Script**. Click on the circle to the right of the box to open the menu and select **ScoreSystem** to add that parameter to all the collectibles.



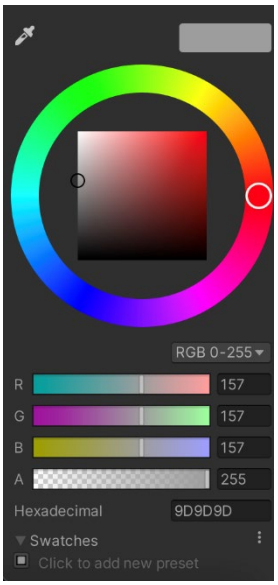
62 Congratulations - your Scavenger Hunt game is ready to play! Go ahead and give it a try!

Customizing the Player Avatar

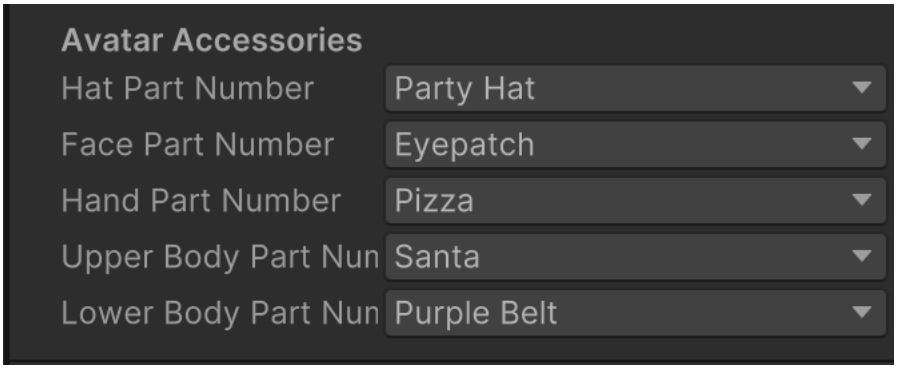
In the Hierarchy panel, select the Player Avatar. Then, in the Inspector panel, scroll down until you find the Avatar Appearance script component.



To change the player's color or skin, click on the appropriate property to open the color selector window. Close the window when you have a color that you like.



To change the accessories, click on any of the menus under the Avatar Accessories heading. Pick the object you want by name (or choose None to leave that section empty).



Save your project and the next time you open your game, it will have the settings that you've selected.