



Imported Assets

Having made a good start on the cursor, now you are probably thinking about adding action objects, some of which will animate. Obviously, the handful of primitive objects available in Unity won't go very far for creating most games, so you'll have to look into one of the more challenging aspects of game creation—importing assets built in other programs. As anyone familiar with creating games or other 3D content can tell you, every game engine or 3D application handles the import process differently. Some handle it more gracefully than others, but there are always going to be issues and gotchas. Adding animations to the mix complicates it even further.

3D Art Assets

Unity was originally designed primarily to create first-person shooters, where imports are most likely characters or static buildings and other structures. A classic point-and-click adventure game, on the other hand, features the animation of a multitude of different objects, many of which animate together to play out complicated sequences as a reward for the player's cleverness in hitting upon the correct solution.

You will start by learning how to handle imported animations, because there are always going to be sequences that are better off imported. Then you'll delve into Unity's Animation view and create a few animations directly in Unity. Characters will be introduced later in the book, because their animation will use the newer Mecanim system and is handled quite differently.

Whether you'll be creating your objects in a Digital Content Creation (DCC) program such as 3ds Max, Maya, C4D, or Blender, or taking advantage of Unity's Asset Store, you will have to start with the import settings. For this book, you are using assets created in 3ds Max and exported in FBX format.

Unity supports the file types of 3ds Max, Maya, C4D, and Blender by converting them internally into FBX format, once they are in the Assets folder. You can still work on them and resave them, and they will be automatically updated on scene load. If you are creating assets for someone else, you should get in the habit of saving as FBX, so they have maximum portability.

For this project, you'll be importing two types of objects: non-animated objects and pre-animated objects. You may think of the former as static objects, such as structures and large environmental objects, but the latter can also be structural as well as animated.

What about materials? Unity attempts to create materials similar to what the object had in the DCC app, but because there are few correlations between regular materials and shaders, you can pretty well count on redoing them all. That being the case, keep materials as basic as possible, as you'll almost always have to rebuild after import, to take advantage of anything other than the apparently default Diffuse shader. And before you go getting all excited about using Diffuse Parallax Reflective Bump shaders on everything, remember that everything affects frame rate—use the fancy stuff sparingly.

■ **Tip** If Unity can't find the textures at the time of import, it will create generic Diffuse materials, and you'll have to set them all up manually once you've loaded the textures. When saving a new DCC file directly to the Assets folder, it's safest to load the textures first, so the materials will be generated with them when the meshes are loaded.

TIPS FOR CREATING YOUR OWN ASSETS

- Name all objects.
- Use naming conventions for your textures. Unity will create only a base Diffuse texture for your objects on import; you will have to add bump, shininess, height, and other specialty maps.
- Two objects using the same material in your DCC app will also share the material in Unity.
- If your modeling application allows, collapse objects to editable mesh, not editable poly, to prevent edge errors. FBX does not support turned edges, so your model could change on export.
- Expect to go back and make adjustments to your files. Real-time renderers and navigation will show errors and mistakes that you may miss in your DCC app.
- When exporting the FBX, make sure to check Animations, if you have them. Y is up in Unity (you can keep the original Z from Max, if you wish, with the latest FBX exporters), and note the scale: it can be changed in Unity, but it is worth noting.
- Keep complex animation hierarchies in orthogonal orientation until after importing into Unity. FBX doesn't support non-orthogonal rotation matrices.
- Unity does not currently support vertex animation, e.g., morph targets. Any animation that affects objects on a vertex basis must be done with bones. Morph target technology is under development, so this tip will eventually be obsolete.
- If you are using 3ds Max, you can use the Resource Collector to copy your textures into the Assets folder before you export the FBX.

Importing the Art Assets

There are several ways to bring assets into your Unity project. You will start by bringing supported file types directly into the project and will eventually learn to import and export Unity Packages.

As with any asset in Unity, once the asset has been imported or added to the Assets folder through the OS, you must only move the file around inside the Project view (expect to be reminded of this regularly—it is crucial!). If you delete it from the Assets folder, it will be deleted for good. If you make your own assets, always have a reference scene in your program's native format stored elsewhere on your hard drive that you can go back to, should you forget and delete something by mistake.

1. Open the CursorControl scene.
2. Delete all but the Control Center, Directional Light, First Person Controller, Terrain, Terrain Boundaries, and WindZone.
3. Save the scene as **ImportedAnimations**.

4. Move it to the Scenes folder in the Project view, if you didn't save it there.
5. Locate the Chapter 7 Assets folder in the downloaded book assets.
6. Copy the Animation Tests folder into the Assets folder in your project folder, via Explorer (Windows) or Finder (Mac).

The folder has both the FBX files and the textures in it, so it should import nicely.

Unity will take a few moments to process the assets, adding internal IDs and mapping out connections between the meshes and materials, as well as generating Unity-specific parameters for the Inspector. When Unity finishes loading, you should see the new Animation Tests folder in the Project view.

1. Open the new Animation Tests folder and inspect the contents, as shown in Figure 7-1.

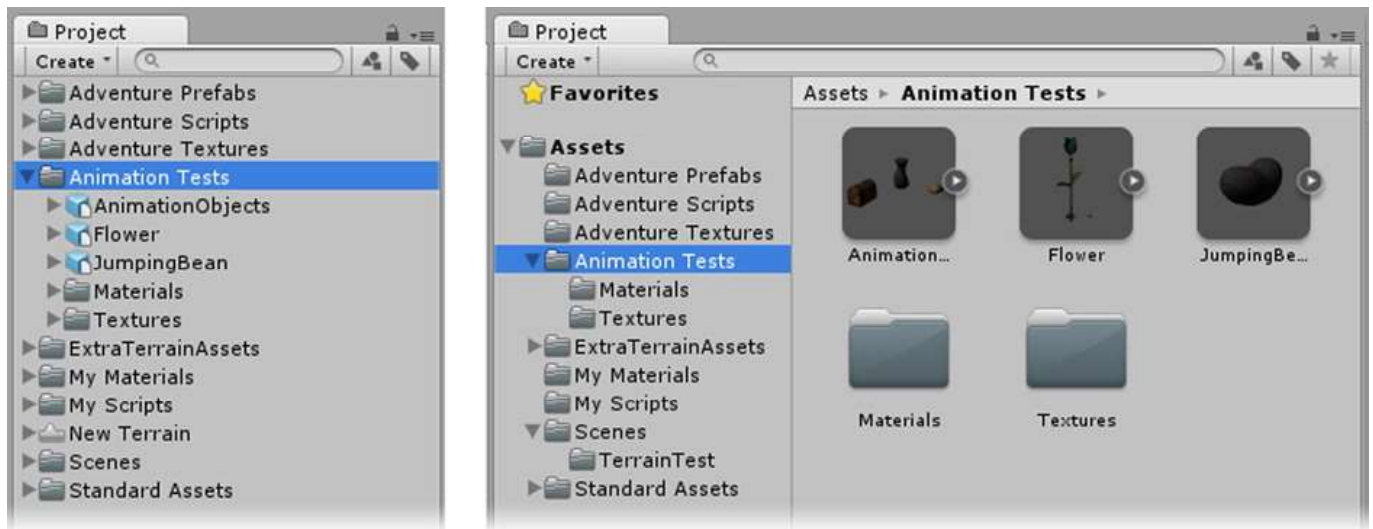


Figure 7-1. The contents of the Animation Tests folder in the One Column Layout (left) and the Two Columns Layout (right)

When importing assets that someone else created, you may wish to get an overview of what you've got. For that, you can switch to the Two-Columns Layout for the Project view. It takes up a lot of screen space but is a quick way to look at what you've just brought in.

■ **Tip** If you've forgotten how to change the layout, right-click over the Project label and select the layout you want. You can adjust the size of the icons from the slider at the bottom of the view. The book will continue to use the One Column Layout for screen shots, but you may use the option you prefer.

Unity has made a prefab to hold each FBX file and, using the preexisting Textures folder, has generated a Materials folder with the materials for each object.

2. Open the AnimationObjects asset (Figure 7-2). In the Two Columns Layout, you will have to click the small arrow on the right side of the thumbnail.

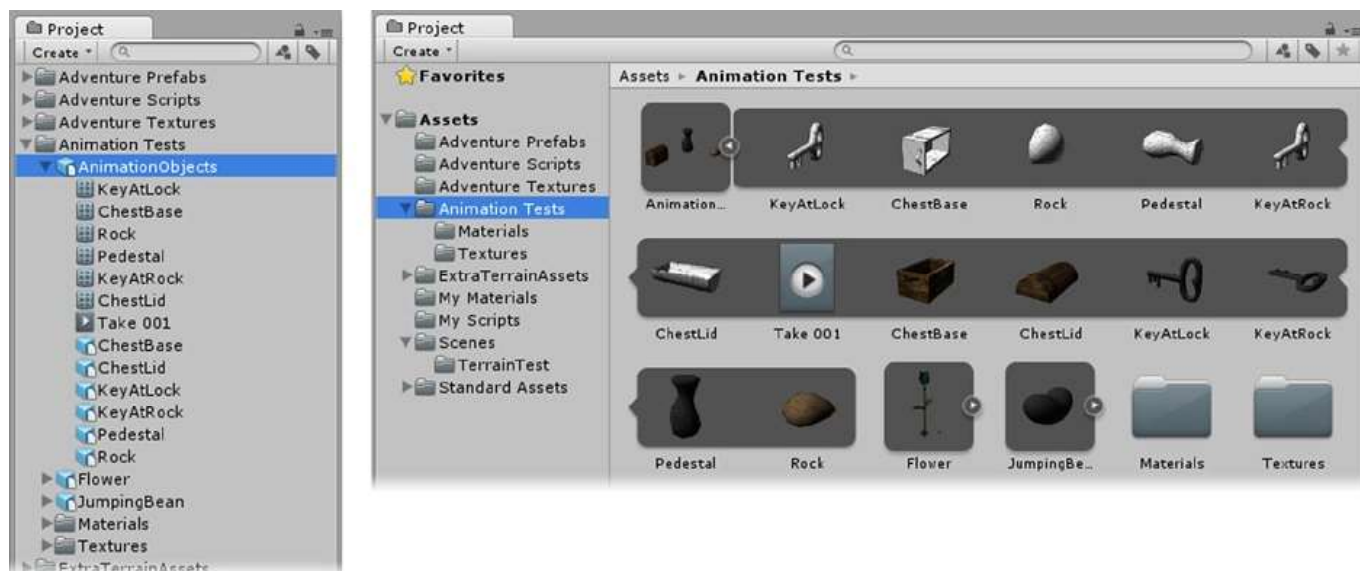


Figure 7-2. The contents of AnimationObjects in the One Column Layout (left) and the Two Column Layout (right)

Inside the AnimationObjects, you'll find a prefab for each individual mesh to hold its properties, the original mesh, and something called Take 001.


3. Click one of the mesh objects, , to see it in the Preview window (Figure 7-3).



Figure 7-3. One of the meshes from the scene

The mesh is shown with edges, and the vertex and triangle (“tris”) count is shown. You can also orbit the Preview window to see the mesh from all directions.

4. Select each item to see its preview in the Inspector.

If an object has animation, you can preview it in the Inspector as well.

5. Select and open the AnimationObjects asset.
6. Select its Take 001 asset—the animation clip.
7. In the Preview window, click the Play button and view the animation (Figure 7-4).

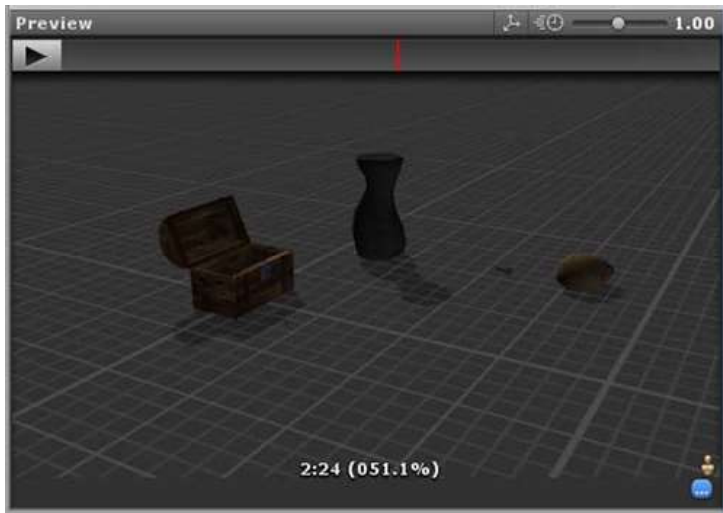


Figure 7-4. The collection of objects run through their animations

8. Check out the Take 001 clips for each of the imported object groups.

The Preview window was designed to show character animation and is only marginally useful for animated inorganic objects. It has a camera that is targeted to the root object of the animations. In the case of the AnimationObjects asset, the group itself does not animate, so the rock quickly moves out of frame, and the small keys barely show.

You might want a bit more practice with the Two Columns Layout. Let's do a bit of experimenting.

1. Right-click the Project tab at the top of the panel; select Two Columns Layout, if you are not already using it.
2. Adjust the spacing and, at the bottom of the second column, move the slider all the way to the right, to see full-sized icons.

Icons are shown for the contents of the Animation Tests folder, including the already opened AnimationObjects.

3. Close the AnimationObjects by clicking its arrow.
4. Experiment with the icon-size slider, to see the viewing options.

■ **Tip** Double-clicking the mesh's GameObject counterparts will open the files in your default FBX app. You may not rename the imported objects in the Project folder.

To keep things less cluttered and allow more screen space for the Scene and Game windows, you will be switching back to the single column. Feel free, however, to switch back and forth on your own, when you find it helpful.

5. Right-click the Project tab and select One Column Layout.

Import Settings

When you bring mesh assets into Unity, they are processed through the importer, for settings that will dictate how the model appears, what system controls its animation, and how the animated clips are defined and interpreted. The three parts of the I are Model, Rig, and Animation.

Import Settings: Model

The first of the import settings is in the Model section. Let's start by selecting one of the newly imported assets.

1. Select the AnimationObjects asset and select the Model section (Figure 7-5).

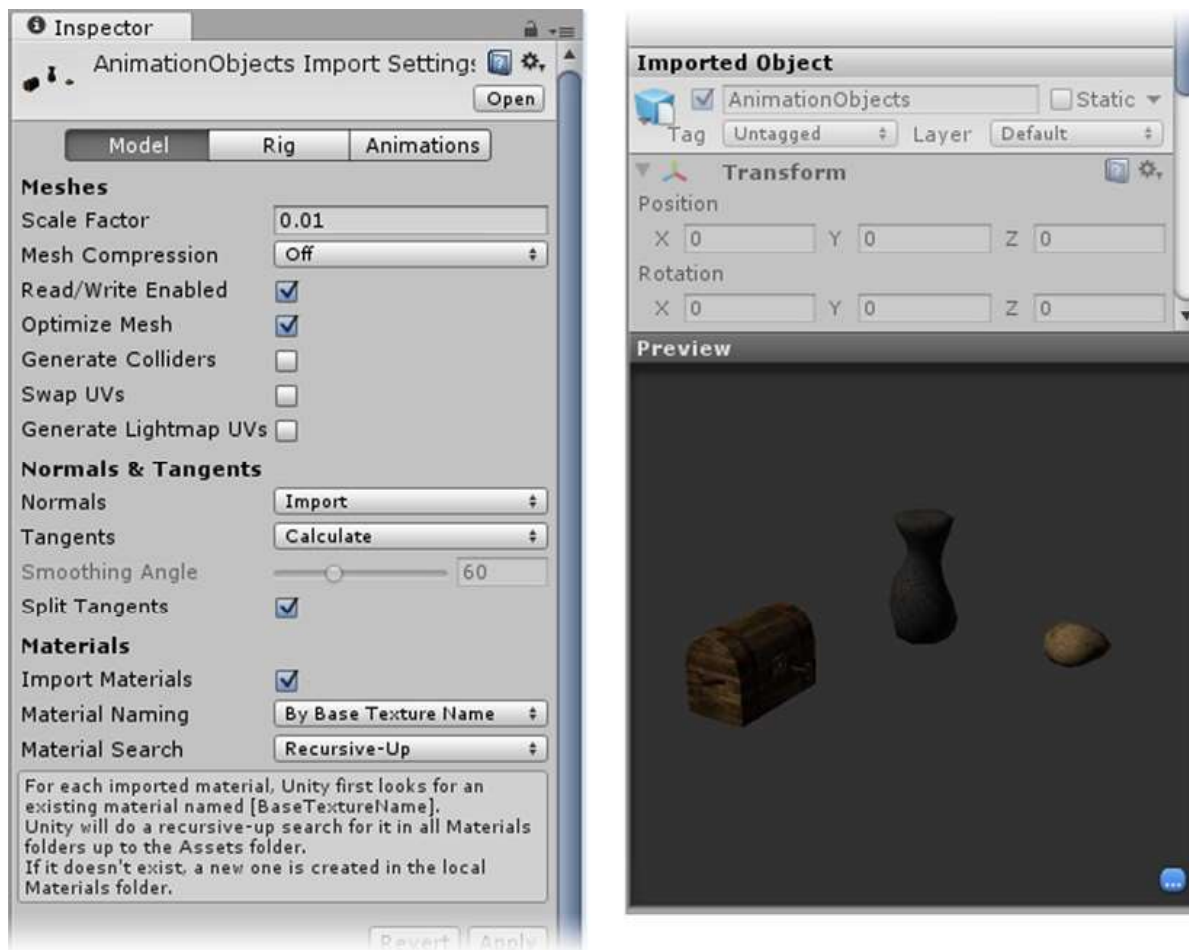


Figure 7-5. The top and bottom of the Model section of the Import Settings

The top setting is Scale Factor, but until you have the objects in the scene, you will have no way to tell whether it is good or not.

2. Drag the AnimationObjects asset into the Scene view, in front of the First Person Controller (Figure 7-6).



Figure 7-6. *The AnimationObjects asset in the scene, scale unknown*

While your own imported meshes are likely to have a consistent scale, you may occasionally come across scenes where that's not the case. Scaling 3D objects in Unity is simple. The unit of measure in Unity is a meter, which is approximately three feet. A newly created Cube is 1 meter cubed and makes a good means of assessing scale. It's not essential that the scale be perfectly correct, only that it be believable.

1. Create a new cube in the scene.
2. Move the cube near the chest to assess the scale (see Figure 7-7).

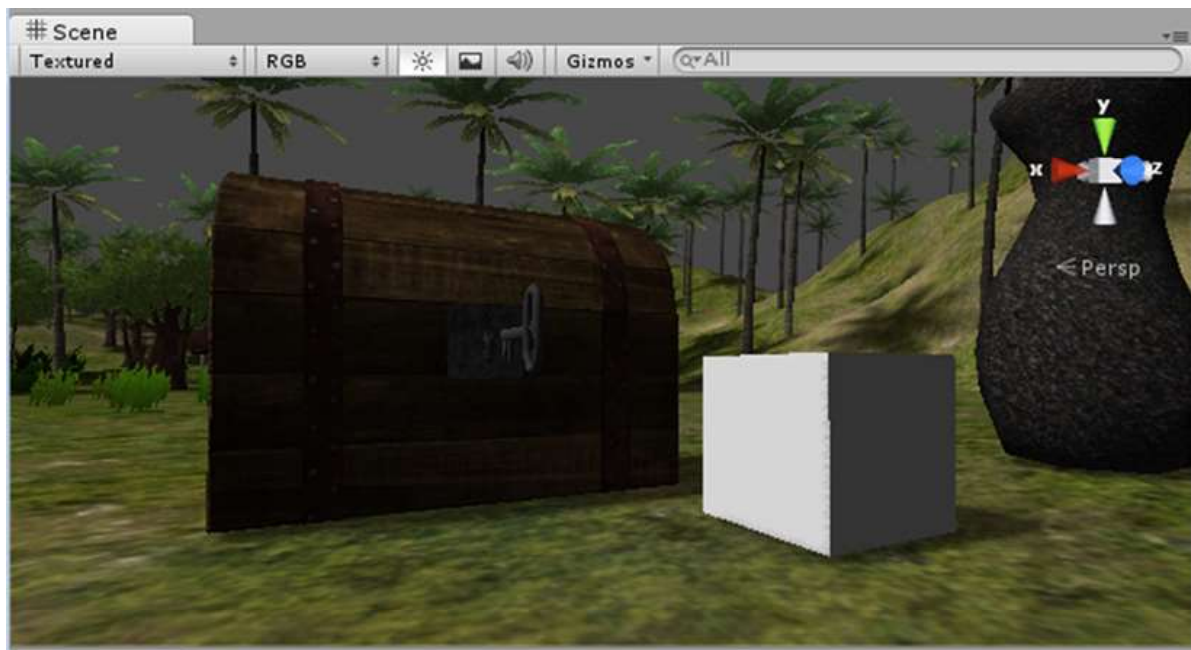


Figure 7-7. Imported object and cube for scale comparison

If the cube is 1 cubic meter, the chest is about the size of a compact car. It looks like the imported objects could stand to be much smaller. While you could scale it in the scene, assets are easier to manage if the scale is adjusted in the imported assets.

3. Select the AnimationObjects in the Project view.
4. In the Inspector, near the top of the Model section, change the Scale Factor to **0.0032** (see Figure 7-8).

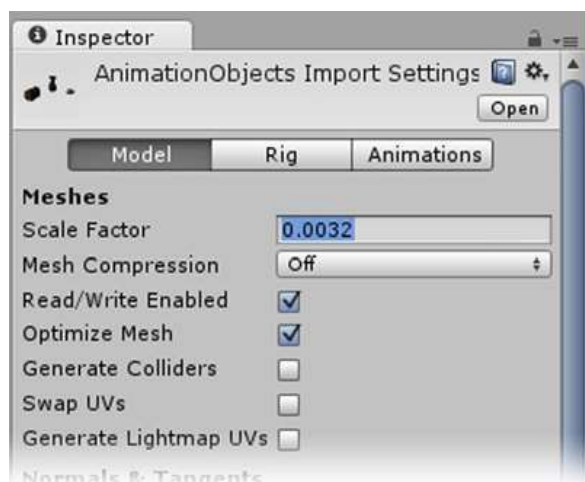


Figure 7-8. Changing the Scale Factor in the Model section

5. Click Apply (the button is about halfway down the panel, on the right side).
6. Change the Scene view to an iso Right view (click the Right label to toggle perspective and iso), to see the results without the distortion of perspective (see Figure 7-9).

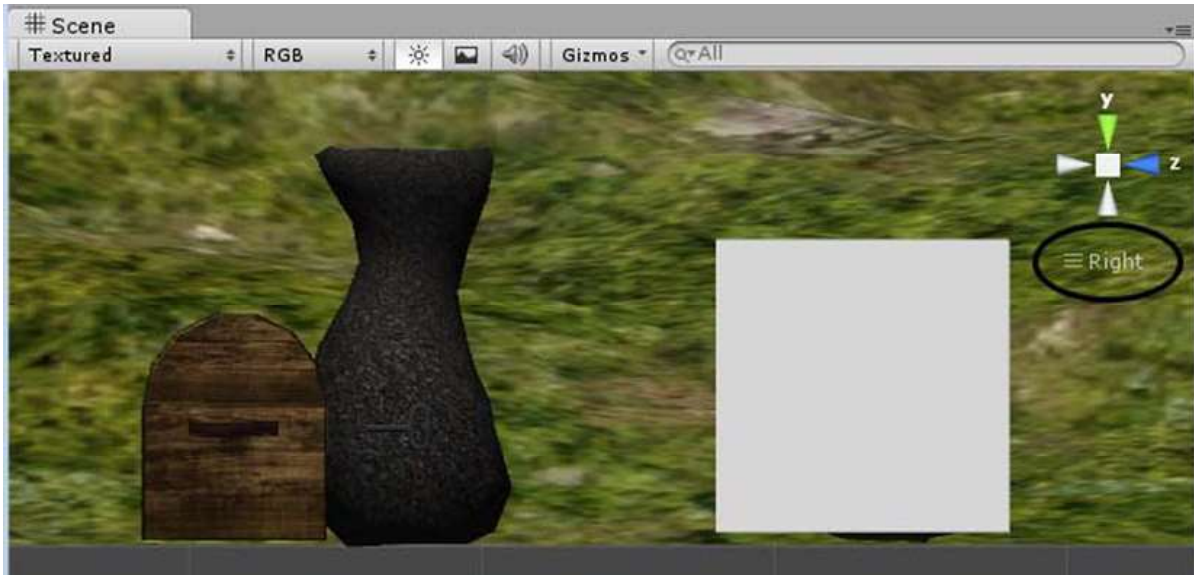


Figure 7-9. Scale comparison, as seen in an iso Right view

7. Rotate the view and click the label to turn it back to a Perspective view.
8. Delete the test cube.

With something more recognizable in the scene, the ground texture looks a bit large. Now is a good time to briefly revisit the Terrain editor and adjust the texture scales.

9. Select the Terrain GameObject, the Paint Terrain tool, and the ground textures.

From Edit Texture, edit the ground textures and reduce the Tiling size to about **8**, then Apply the change. Figure 7-10 shows the ground before and after editing.



Figure 7-10. The ground with the default 15x15 scale (left) and the new 8x8 scale (right)

■ **Tip** In a fast-paced game, the large tiling would be less obvious, but in a classic point-and-click, the player is more likely to be focused on the immediate surroundings and will notice poorly scaled textures.

The final test of any adjustment is to go into Play mode and see how it looks “in game,” as the player navigates the scene.

1. Toggle on Scene Lighting in the Scene view and move and rotate the group or the First Person Controller until the imported objects can be clearly seen in the Game view.
2. Click Play.
3. Drive up to the objects and look around them (Figure 7-11).

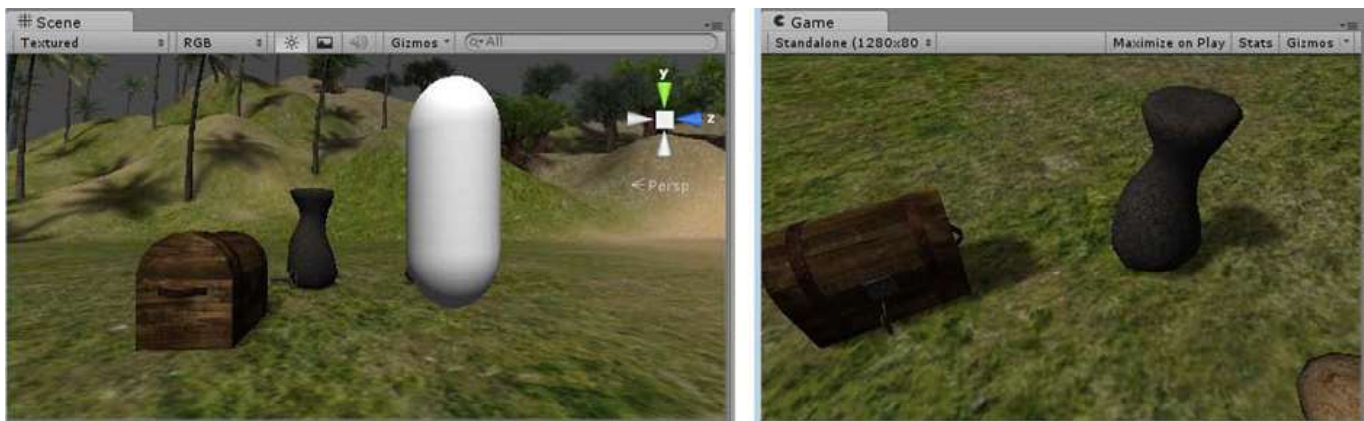


Figure 7-11. The view in the Scene view (left) and from the First Person Controller (right)

Either the objects have to be bigger, or you will have to adjust the camera’s field of view.

In games, scale is not always exact. Often you’ll have to increase the scale of an object to make it more playable. In our case, however, it turns out the camera field of view is more suited to a first-person shooter that requires more peripheral vision than for an adventure game where small objects need to be noticed.

1. Exit Play mode.
2. Open the First Person Controller.
3. Select the Main Camera.
4. In the Inspector, change the camera’s Field of View to **32**.
5. Examine the objects again.

As Figure 7-12 shows, now you can get a good view of the objects when you are close.



Figure 7-12. *Reduced field of view on the First Person Controller's camera*

6. Stop Play mode.
7. Make the change to the camera permanent.

After you've finished with your test objects and have more of the scene completed, you'll probably want to adjust the camera's Field of View again, but while you are setting things up, you may as well make it easy to access the objects.

1. Select the Main Camera.
2. Set its X Rotation to about **12**, to get a better view of the objects for when they animate (Figure 7-13).

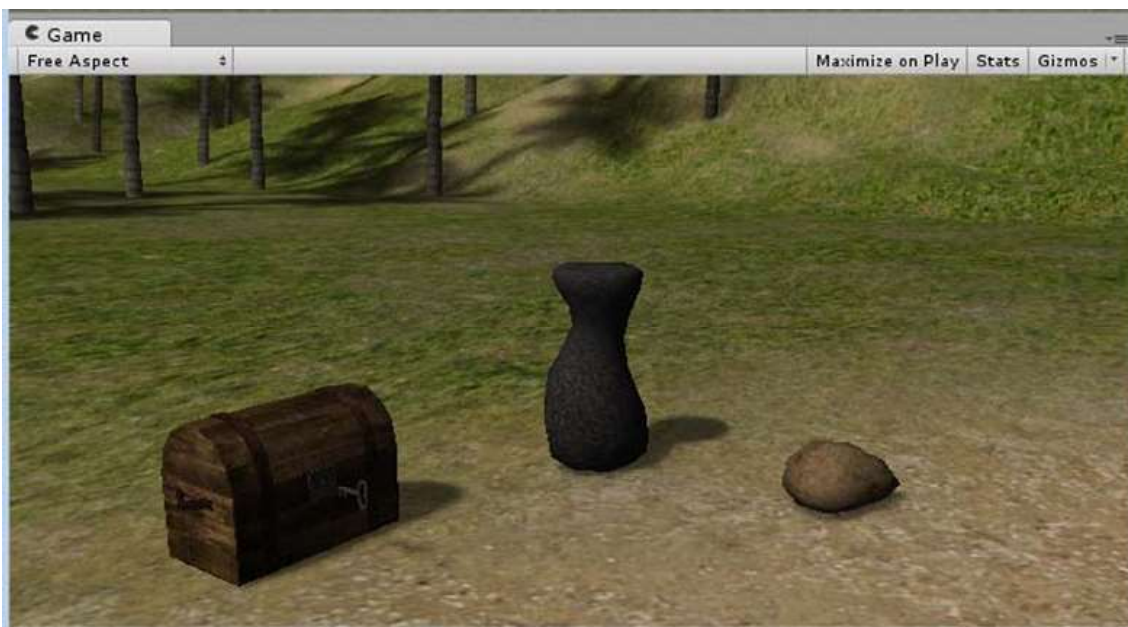


Figure 7-13. *The objects ready to view*

3. Adjust the AnimationObjects group and First Person Controller in the Scene view, so the objects are visible in the newly adjusted Game window. Do not move the individual objects.
4. Set the Game view back to Free Aspect.

Import Settings: Rig

With Unity's new Mecanim system, you have several choices to make for your animation. Eventually, the original animation system will be merged into Mecanim, but until then, you will have to wade through the lot. Mecanim was originally designed to control Humanoid characters. This precluded things such as tails, wings, and ponytails, so they added a Generic option. Then they discovered that there were cases that didn't lend themselves to Mecanim as it stands, so they re-exposed the Legacy option in the import settings.

If that sounds confusing, it's because Unity made a decision to release features throughout the point-release cycle. Instead of waiting for relatively finished features to be unveiled only during major releases, you can expect features to be altered, refined, and improved at any point release. Always be sure to check the Unity forum to learn how to deal with changes. This book has a thread on the Teaching forum using its title.

The next section of the Import Settings is the Rig section. As a default, imported objects are assigned the Generic Animation Type.

1. Select the AnimationObjects in the Project view and select the Rig section (Figure 7-14).

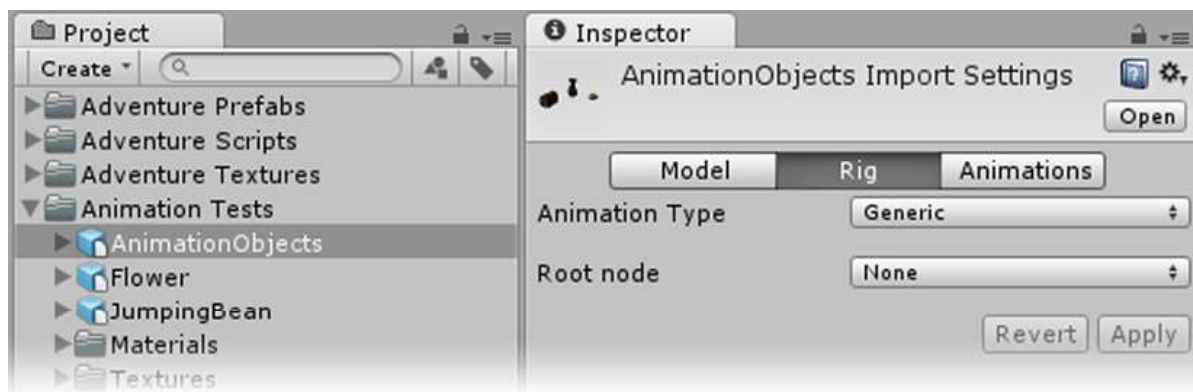


Figure 7-14. The Rig section of the Import Settings

2. Click the Animation Type drop-down (Figure 7-15).

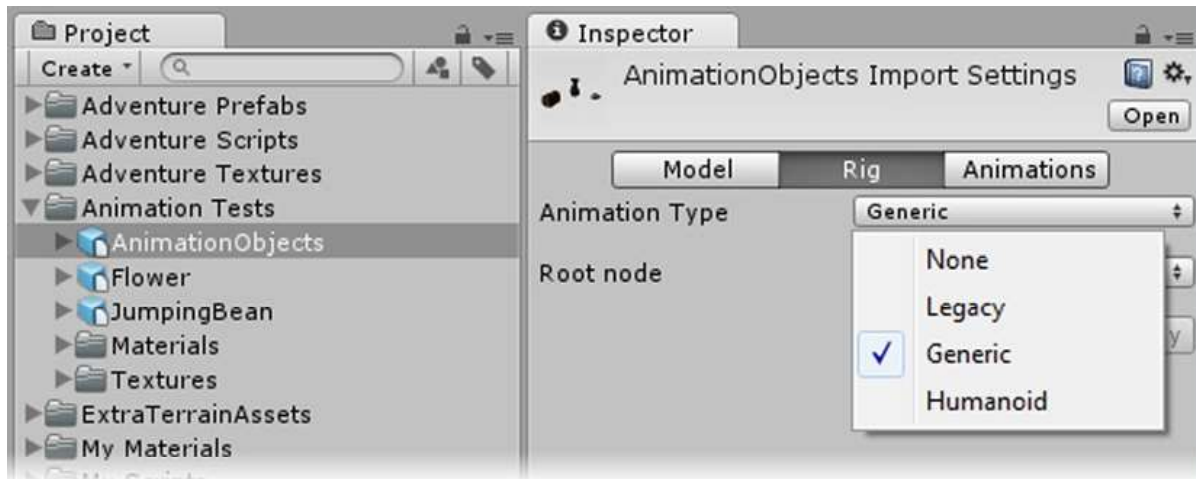


Figure 7-15. The Animation Type choices

The options are None, Legacy, Generic, and Humanoid. The objects, or at least some of them, have to animate, so None is not the right choice. The objects are obviously not Humanoid, so that one is out as well. So the two choices are Legacy and Generic.

Generic uses the Mecanim system and adds an Animator component.

1. Select the Animation objects in the Hierarchy view and check out the Animator component in the Inspector (Figure 7-16).

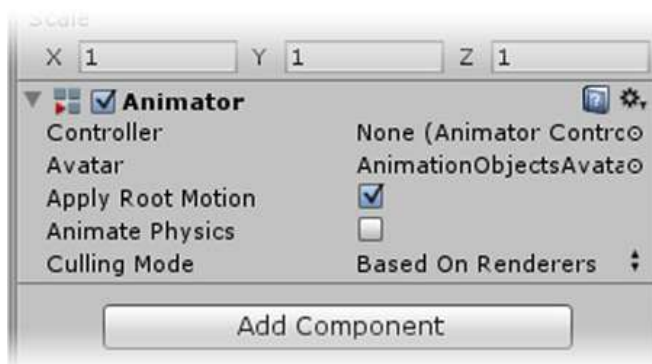


Figure 7-16. The Animator component

Half of the Animator component appears to be character-related. But the two important things to know about the Animator component/Mecanim system are that there is no way to test the animation in the scene when the objects are in place and that it has built-in blending. Blending moves an object from one animation smoothly into another. A typical example is a character in walk animation smoothly blending into a run animation. The animations, including the blends, are viewed in the Preview window.

Both issues, while advantageous when handling characters, will be detrimental to setting up the simple mechanical animations used on most of the action objects. To set up the action objects, you will want to observe the animations in the scene, to make sure they are correctly placed in respect to the animations they hold. The rock, for instance, needs to animate into a clear spot, without intersecting anything else on the way. The objects also have fully contained animations. Not only is blending unnecessary, most objects only have one animation. Blending also carries

a lot of overhead, both in scripting and setup. Bottom line: Mecanim is not suited for the simple animation you need for most of the action objects.

2. Select the AnimationObjects asset in the Project view.
3. Change the Animation Type to Legacy and click Apply.
4. Open the Generation drop down to view the choices (Figure 7-17).

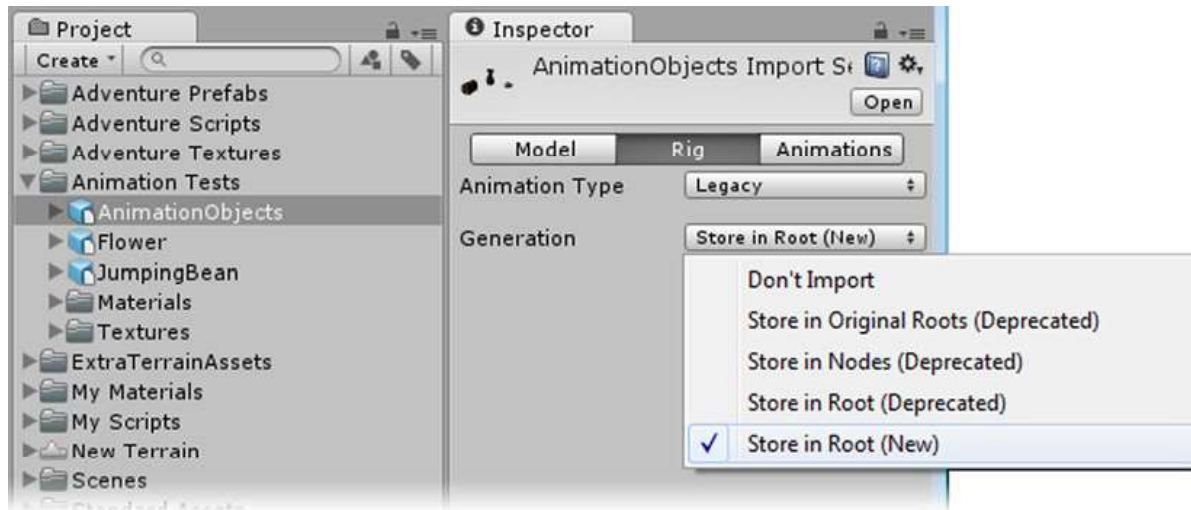


Figure 7-17. The root Generation type

As you can see, other than Don't Import, most of the options are deprecated. The default, Store in Root (New), is the preferred choice. This option puts the animation on the root object. For the current asset, this means the Animation component will be put only on the parent object, the AnimationObjects group. The Take 001 clip you saw earlier will be split up to define the animation clip for each individual object, but they will all be called on the parent object. One downside to this option is that you can only play one clip at a time. Triggering another while one is playing will cancel the clip currently playing. When setting up this type of animation, each behavior must have a unique place on the time line.

When you switched to the Legacy Animation Type, the Animator component was replaced by the Animation component. Let's take a look at it next.

1. Select the AnimationObjects group in the Hierarchy view.

It now contains an Animation component (Figure 7-18).

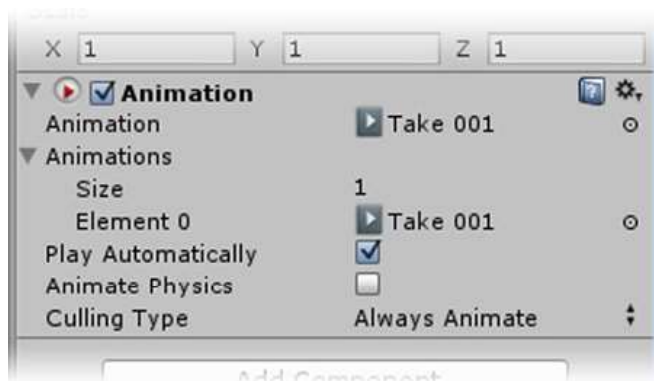


Figure 7-18. The Animation component

The Take 001 clip is the current default animation clip, because it is the only clip so far. The best part of this component is the Play Automatically option. This will allow you to see the animation in the scene, without setup or code, while you arrange the scene.

2. Click Play and watch the Game or Scene view to see the animation (Figure 7-19).

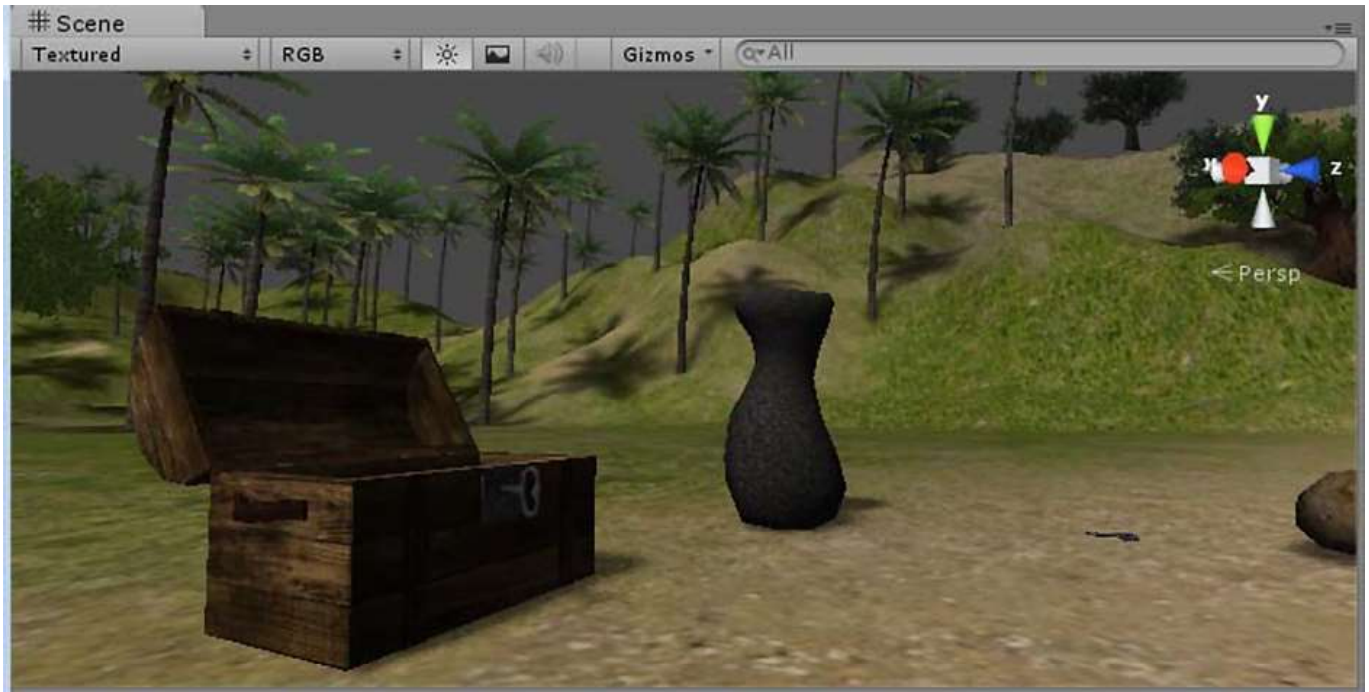


Figure 7-19. The animations playing automatically at startup

3. Stop Play mode.

Parenting

One of the most important concepts to understand in 3D animation is parenting. *Children inherit the transforms of their parent.* Transforms are translate (move), rotate, and scale. Inside the parent group, AnimationObjects, the individual objects have their own animations that are enacted relative to the parent group. No matter where you place the parent group, all the animations happen as expected.

Let's try a little experiment and see what happens if you move an object from inside the AnimationObjects group.

1. In the Hierarchy view, select the rock and move it back, away from the key (Figure 7-20).

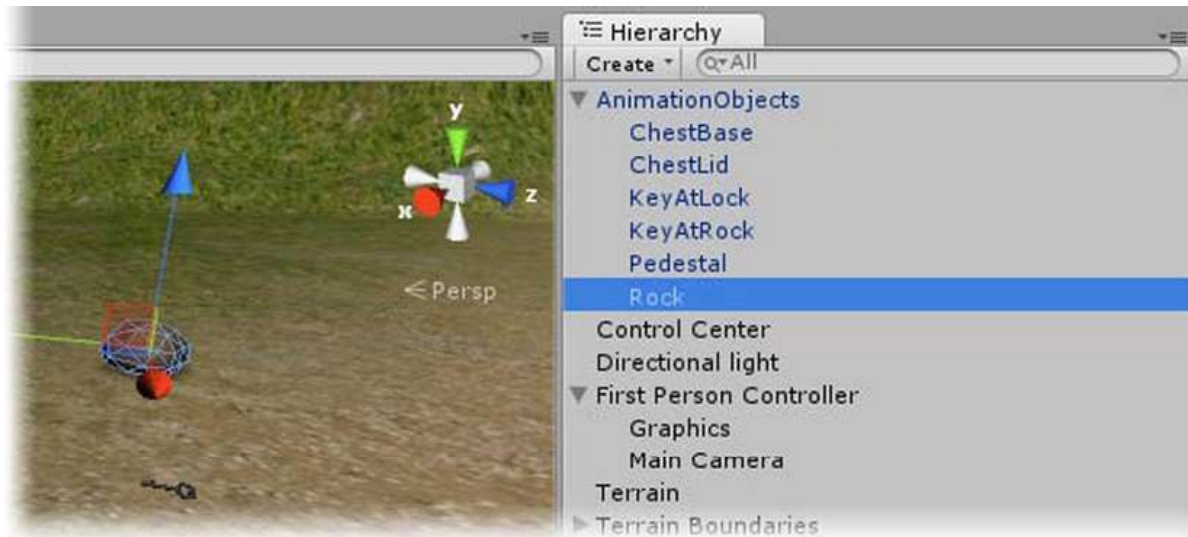


Figure 7-20. The rock moved away from the key

2. Click Play and watch what happens.

When it comes time for the rock to animate, it jumps back to animate from the original location relative to its parent, the AnimationObjects group.

You're probably wondering what would happen if you just removed it from the group.

3. Stop Play mode and do a couple of undo's to get it back to its original position.
4. In the Hierarchy view, drag the rock out of the parent group, and drop it below the other objects in the list.
5. Agree to the Losing Prefab warning.
6. Click Play.

Nothing happens, because it is no longer under the control of the group's Animation component. Even if you copy and paste the component onto the rock, the association has been lost. Looks like you'll need a different object for this test.

1. Select the JumpingBean from the Animation Tests folder in the Project view.
2. In the import settings, Rig, change the Animation Type to Legacy and click Apply.

Once you've switched from Generic to Legacy, the Take 001 will no longer show the animation in the Preview window. The message says you need to drag a model into the view. This means you need the whole asset, not just its mesh component (Figure 7-21).

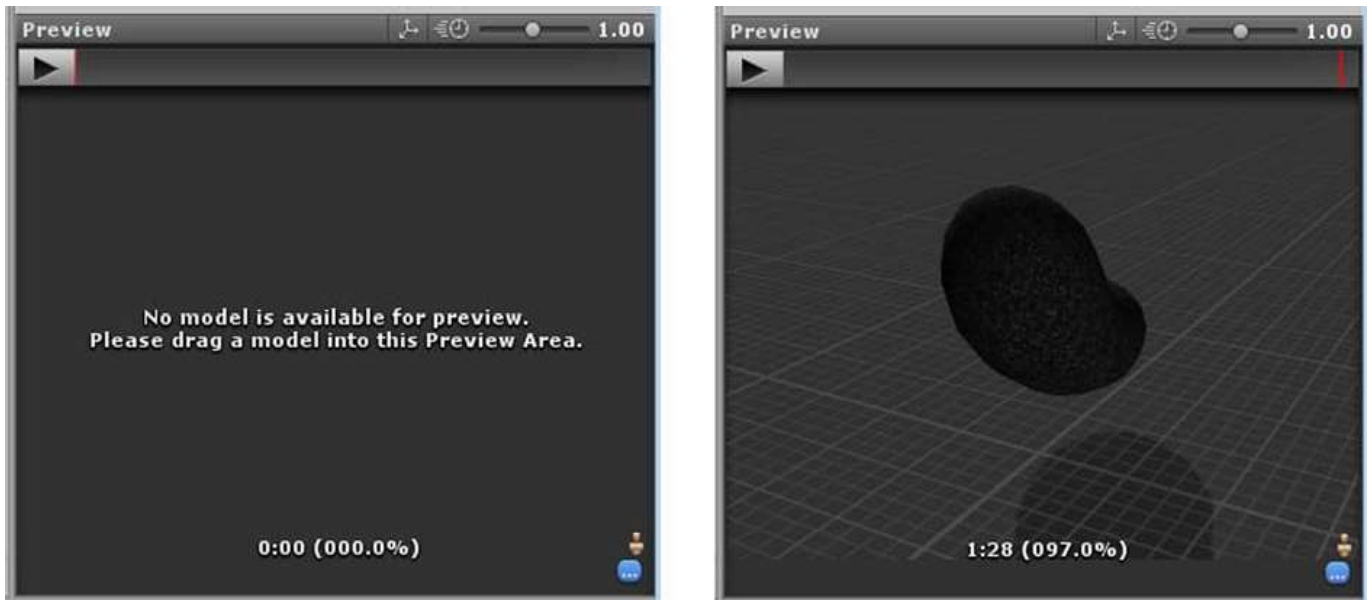


Figure 7-21. Using the Preview with a Legacy animation clip

■ **Tip** The next time you want to preview an asset's animation, the bean will still be loaded, so you will have to drag the new asset into the window before its animation will show.

3. Drag the bean into the scene near the other objects.

In the Hierarchy view, you can see that it is a single object.

4. Click Play.

It completely disappears! If you double-click it in the Hierarchy view, you will find it below the terrain, somewhere close to 0,0,0, where it was originally animated in the original file. Without a parent, an imported object is automatically parented to the scene itself, so just like when you moved the rock, the bean jumped back to its original location to do its animation.

1. Delete the JumpingBean from the scene.
2. Drag a new one into the Hierarchy view, rather than directly into the Scene view.
3. Double-click it to find it in the Scene view.

Note that it is fairly close to 0,0,0.

4. Now create an Empty GameObject and name it **Bean Group**.

It is created at the focus spot—the same location as the bean.

5. In the Hierarchy view, drag the JumpingBean object onto the Bean Group.
6. Focus back on the AnimationObjects group, then use Move to View to move the Bean Group into place.
7. Click Play and watch the Bean Group.

This time the bean animates as expected. As long as you move the new parent object, not the object with the animation, everything will work correctly. So, the solution to an animated object that does not have a parent on import is to put it inside a group before you move it.

In this type of genre, it's not unusual to have several independently animating objects imported as a single grouped asset. With the AnimationObjects group, you can see that the key's position relative to the chest it unlocks is crucial. The rock, and key it covers, is a different case. While they should probably stay near the chest, you might find you have to make some minor adjustments to fit them into the scene just right. You now know you can't just move the rock around, so you need a solution that will let you separate animated objects from their parent group, without disrupting their animations. Turns out, it's easy. All you have to do is make a duplicated group and delete all the extra objects in it.

Let's start by getting the AnimationObjects group back to "stock."

1. Select the AnimationObjects group and, at the top of the Inspector, in the Model line, click Revert.
2. Delete the Rock that originally came out of the group.
3. Select the AnimationObjects group and use Ctrl+D to duplicate it.
4. Name the new group **Rock Group** and delete all but the Rock and KeyAtRock objects inside it.
5. Select the original group and delete the Rock and KeyAtRock.
6. Move the new Rock Group to a new location and click Play.

This time, the animation is still intact.

Prior to 4.0, Unity had an option to store imported animations on the object roots. This meant you could import a scene full of animated objects all together, yet have control of them as individuals. The option is still there, but it is one of the "deprecated" ones and is painful to set up. Hopefully, a new solution will eventually be developed to replace it. Until that time, you should now have a pretty good understanding of how to manage assets with multiple and distinct animating objects.

Import Settings: Animations

Now that you've got some objects in the scene and animating, you will need a way to both separate the animations for each object and to have them play on cue. The first step is to break the generic Take 001 into separate behaviors or animation clips. To begin, you will want a list of the frames used by each object's behaviors, so you can define the clips.

Table 7-1 shows the time segments for the various objects and the names you will use for the clips.

Table 7-1. *Object Behaviors in the Animation*

Clip Name	Start Frame	End Frame
rock move	0	30
key insert	31	60
chest open	61	90
chest close	91	110
key remove	121	150
chest rattle	151	163

■ **Tip** In scenes with multiple objects imported, name clips with the object name first and the behavior second, such as *chest open*, *chest close*, or *chest rattle*. This will make clips easier to locate in the browser. Another naming convention that you will be following is to use all lowercase for the legacy animation clips.

1. Select AnimationObjects in the Project view and expand it.
2. In the Inspector, click the Animations section of the Importer (Figure 7-22).

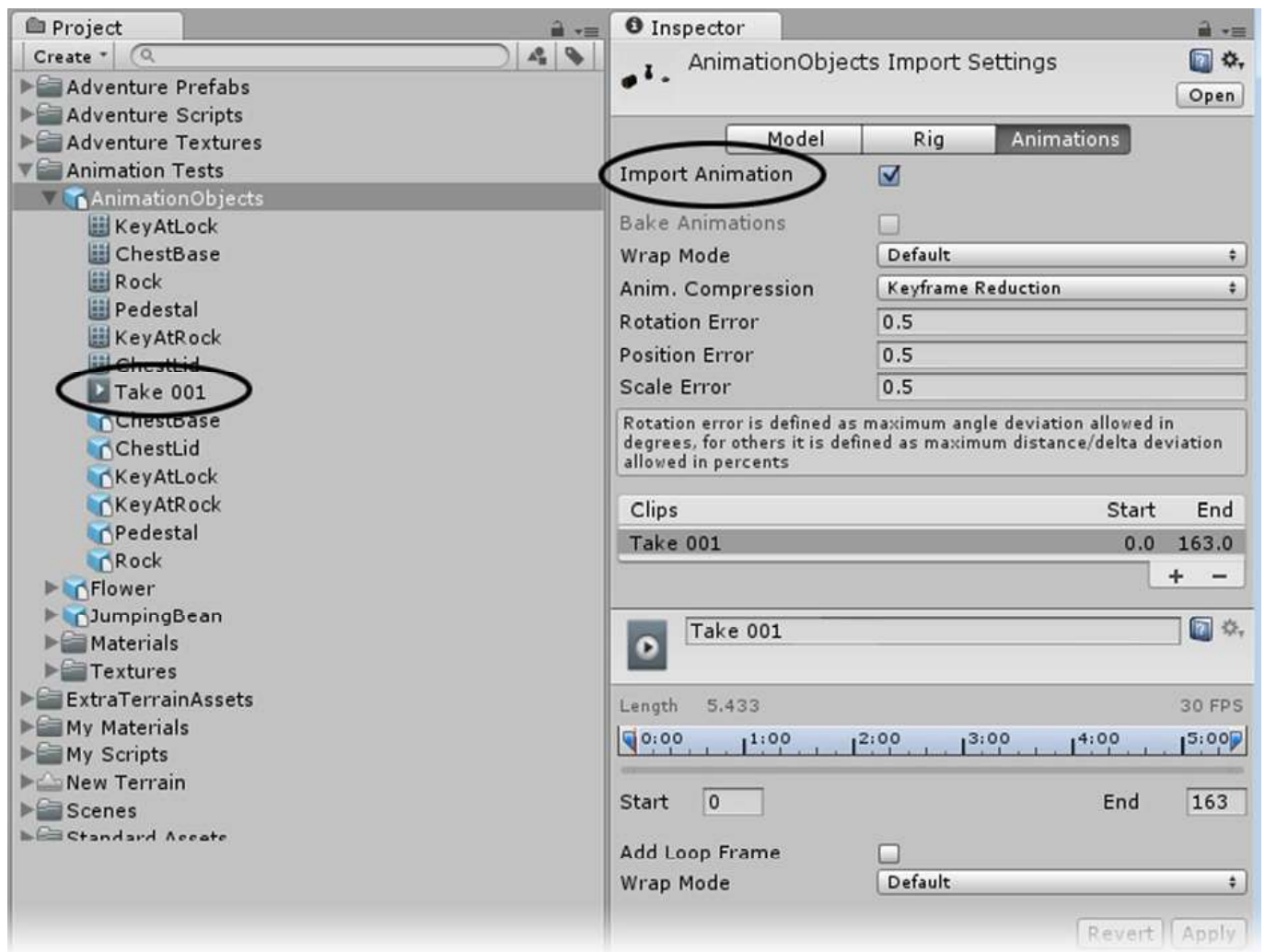


Figure 7-22. The animation section and the default clip

3. Note the location of Take 001, then, near the top of the Inspector, uncheck Import Animation.
4. Click Apply.

Take 001, which represented the animation, no longer appears in the Project view, and the Animation section is empty.

5. Click Play.

The AnimationObjects' animations no longer exist.

Just as it says, animation is not imported for the scene. Use this setting for things such as buildings, landscaping objects, and other static or non-animating objects.

6. Re-check Animation.
7. Click Apply, then click the Animations tab once, to load the clip information again.

The Take 001 clip is reinstated.

8. Select the AnimationObjects in the Hierarchy view and check out the Animation component (Figure 7-23).

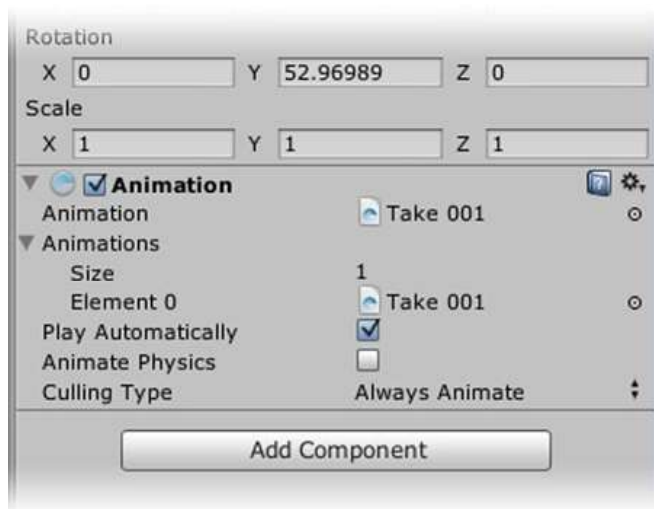


Figure 7-23. The Animation Component showing the imported clip

The Animation array has one element, and the default Animation is Take 001.

1. Select AnimationObjects in the Project view again.
2. In the Inspector, change Wrap Mode to Once (Figure 7-24).

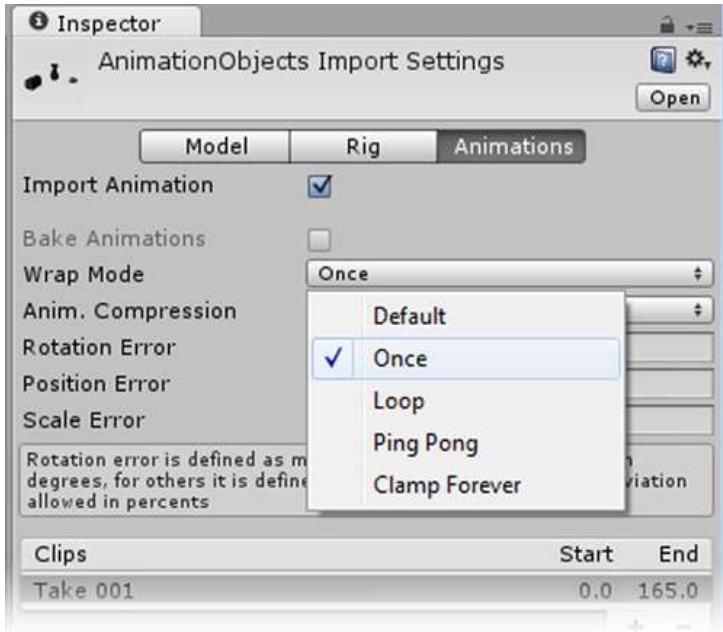


Figure 7-24. The Wrap Mode setting that will be the default for all of this asset's clips

3. In the Clips list, select the Take 001 clip and, next to the Play arrow, rename it **all** (Figure 7-25).

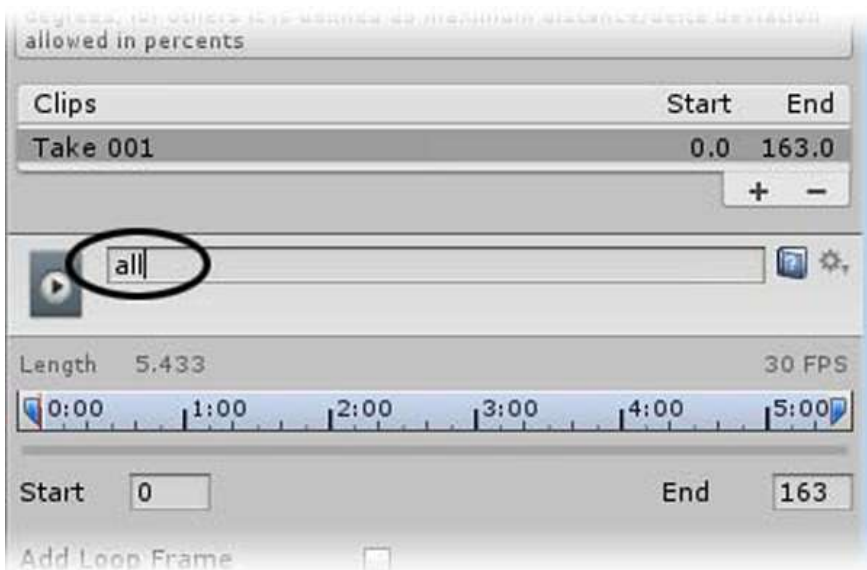


Figure 7-25. Renaming a clip

As soon as you press Enter, the Take 001 name will update. There are a few things happening here that bear explaining. By keeping the clip of the entire animation intact, you keep any easy means of testing a complicated sequence of animations. Also, in case you were wondering, if you look above the time line, it shows the length of the clip, 5.433 seconds, and the frame rate used when creating it, 30 frames per second. Unity quite happily uses whatever frame rate was used in the creation of the imported animation.

4. In the Clips list, click the + to generate a new clip.
5. Name it **rock move**.
6. Below the blue time line, leave Start at 0 and change End from 163 to **30**.
7. Drag the AnimationObjects asset into the Preview window.
8. Right-click the Preview's title bar to float it, so you can make the window larger.
9. Click the Play arrow to see the animation for the clip play.
10. Adjust the time the clip takes to play out by changing the 1.00 to 2.00 (faster) and 0.1 (slower). Return it to 1.00 when you are finished.

The Preview doesn't have an option for a wrap Once, so you will have to drag the time indicator manually to see if the time segment of 0-30 is good. You can re-dock the Preview window by clicking the × at the upper right.

With the first regular clip now defined, let's add the rest of the clips, according to Table 7-1.

1. In the Clips list, click the + to generate a new clip.
2. Name the new clip **key insert**.
3. Set its Start to **31** and its End to **60**.
4. Repeat the procedure for the remaining clips, according to Table 7-1.
5. When you have finished, click Apply (see Figure 7-26).

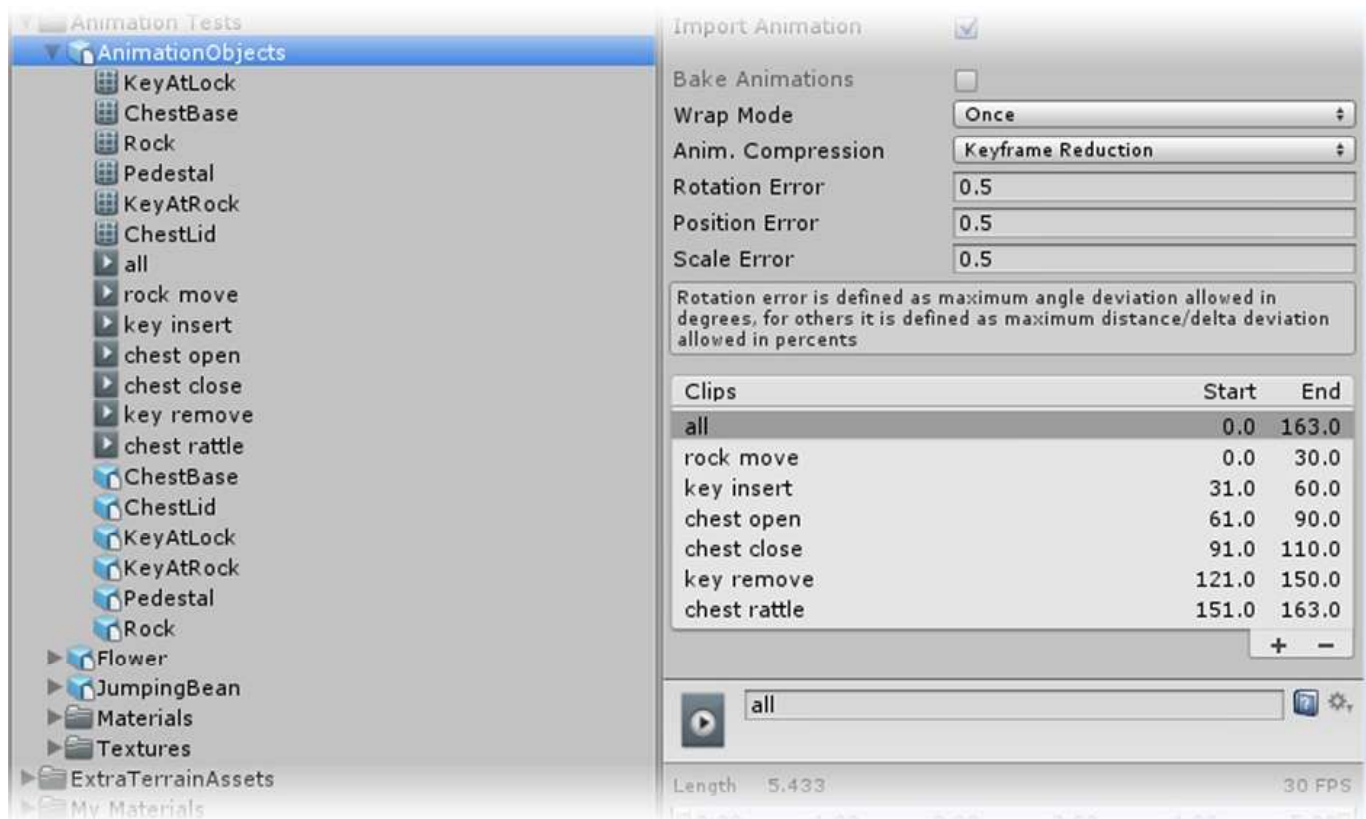


Figure 7-26. The newly defined clips in the Project view and in the Inspector

Having added several clips and renamed the original Take 001 for the AnimationObjects group, we ought to check back with the objects already in the current scene to see what, if any, changes were made in the Animation component.

1. Select the AnimationObjects in the Hierarchy view.

The Take 001 has been changed to *all*, but the new clips were not loaded because the prefab was broken.

2. Click Revert in the Inspector to load the new clips into the Animation component (Figure 7-27).

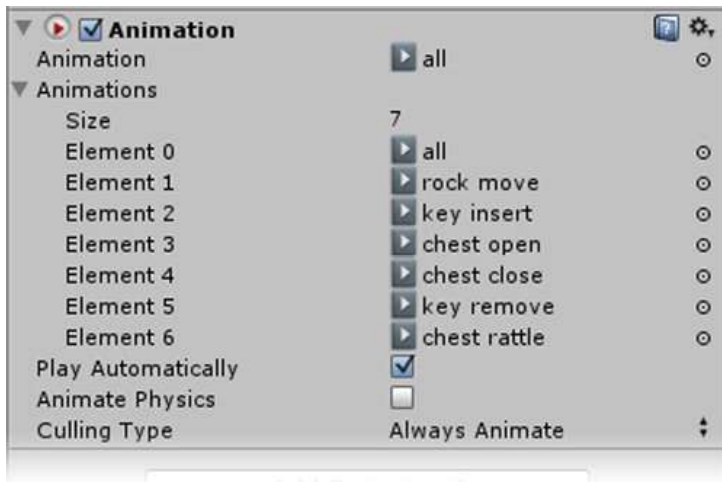


Figure 7-27. The updated clips in the Animation component

3. Load one of the new clips in as the default Animation through the browse icon.

Unfortunately, you can't just drag the one you want from the Animations list, hence the emphasis placed on naming and naming conventions for the clips. You can, however, click the existing clip. It will be briefly highlighted in the Project view, allowing you to quickly locate the one you really want and drag it into the field.

4. Click Play and watch your chosen animation.
5. Exit Play mode.

Now that the clips are safely added, you can delete the Rock and KeyAtRock from the group, once again breaking the prefab. The takeaway here is that you should usually set up the clips before doing much in the scenes. Since things tend to change, that won't always be possible, so you will get a chance to manually fix the Rock Group.

1. Delete the Rock and KeyAtRock from the AnimationObjects group.
2. Select the Rock Group.
3. Drag the rock move clip from the Project view onto the existing all clip in the Animations list.

Rather than replace the existing animation, the list Size is increased and the *all* clip is bumped down an element.

4. Set the default Animation to rock move.
5. Click Play.

You probably didn't notice anything different, because the rock move uses the first 30 frames of the whole animation. If it had started later in the original clip, you would now see it play immediately, instead of at its original position on the time line.

6. Exit Play mode.

As a final cleanup, even though you are not going to use the `JumpingBean` in the game, you should probably rename its `Take 001` clip to remove one more generic `Take 001` from the project before you move on.

7. Select the `Jumping Bean` in the Project view.
8. In the Inspector, Animation section, rename `Take 001` to `bean jump` and click `Apply`.

Importing Complex Hierarchies with Animations

So far you have seen fairly simple animations on an asset with several distinct members. The remaining import is the `Flower`. It has its own complicated bone system, as well as FK (forward kinematics) animation. Although the flower will not be hopping around the scene trying to eat unsuspecting players, it is controlled much as a character is, in that the Animation component is on the root or parent and clips animate all the children beneath the root.

1. Select the `Flower` asset in the Project view.
2. Change its Animation Type to `Legacy` in the Rig section of Import Settings.
3. Add the clips shown in Table 7-2.

Table 7-2. *New Flower Behaviors*

Object, Behavior	Start Frame	End Frame
flower revive	0	124
flower idle	125	200

4. Just above the `Apply` button, set the `revive` clip's `Wrap Mode` to `Once` and the `idle` clip's `Wrap Mode` to `Loop`.
5. To ensure a smooth looping transition for the `idle` clip, check `Add Loop Frame`.
6. Click `Apply`.
7. Drag the `Flower` into the scene and move it so that it is visible in the Game view.
8. Click `Play` to see its entire animation.
9. Change its default animation to the `idle` clip and click `Play`.

The `idle` behavior loops. It turns out that the flower is just as ugly revived as wilted (Figure 7-28).



Figure 7-28. *The flower in its idle behavior*

Setting Up Materials

As mentioned earlier in the chapter, Unity imports materials only on a very limited basis, and most of the action objects, at least, deserve better than the default material. This means you'll definitely have to spend some time customizing and improving the defaults that are created on import. Unity uses shaders exclusively, so you'll be able to get a lot of rather nice effects once you understand how they can be manipulated.

Let's look, then, to see how the objects came in. Because the textures were brought in at the same time, most of the objects will have had a default material created from their Diffuse texture. The default material uses a simple Diffuse shader that has only a Base texture and a Main Color that can be used to tint the texture map. Naming is from the texture first, and the original material second, by default. If you wish, you can change the naming convention in the Model section of the Importer, under Materials, Material naming.

You will be going through the imported objects one at a time and will “doctor” them up a bit.

The Keys

The two keys, a couple of wrought iron types, originally had only a bump map. Because they had no diffuse, Unity named the material Key, after the original material (Figure 7-29).

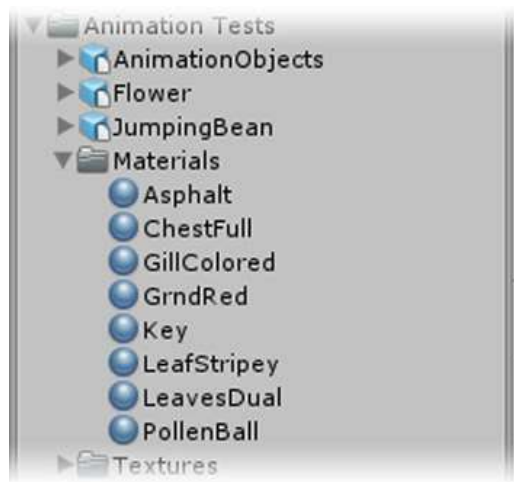


Figure 7-29. Materials generated for the imported objects

In 3ds Max, where the meshes were made, the bump uses a texture called Hammered.jpg. In Unity's shaders, you will use a Normal Map instead.

1. Select the Hammered texture from the Textures folder.
2. Check it out in the Preview window in the Inspector (Figure 7-30).

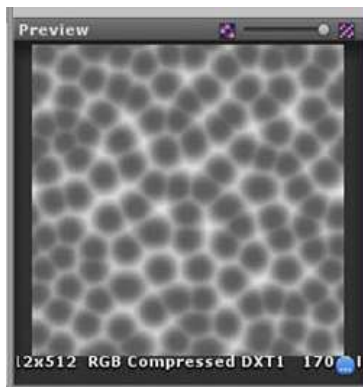


Figure 7-30. The Hammered texture in the Preview window

The texture has already been converted to a DXT1 texture format, and Mip Maps have been generated for it. Unity converts all non-dds formatted texture to dds on import.

Unity can automatically turn it into a normal map for you, so you will need to duplicate it first.

3. Use Ctrl+D (or Cmd+D on the Mac) to duplicate the Hammered texture.
4. Select Hammered 1.
5. In the Texture Importer, change the Texture Type to Normal map.
6. Change the Bumpiness to about 0.15 and the Filtering to Smooth.
7. Click Apply.

The map is now the distinctive Cyan/Magenta colors of a normal map (Figure 7-31).

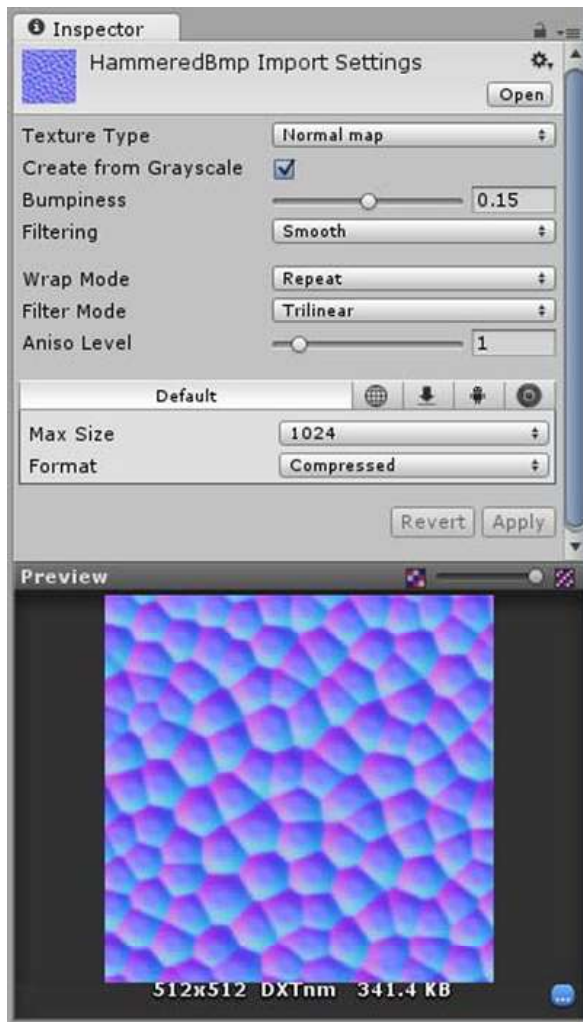


Figure 7-31. The Hammered texture converted to a normal map

8. Rename the texture to reflect its new function: **HammeredBump**.
9. Select the Key material.
10. At the top of the Inspector, click the Shader drop-down and change it from Diffuse to Bumped Specular, the second option from the top.
11. Leave the Base map as None and drag and drop the HammeredBump onto its Normalmap thumbnail (Figure 7-32).

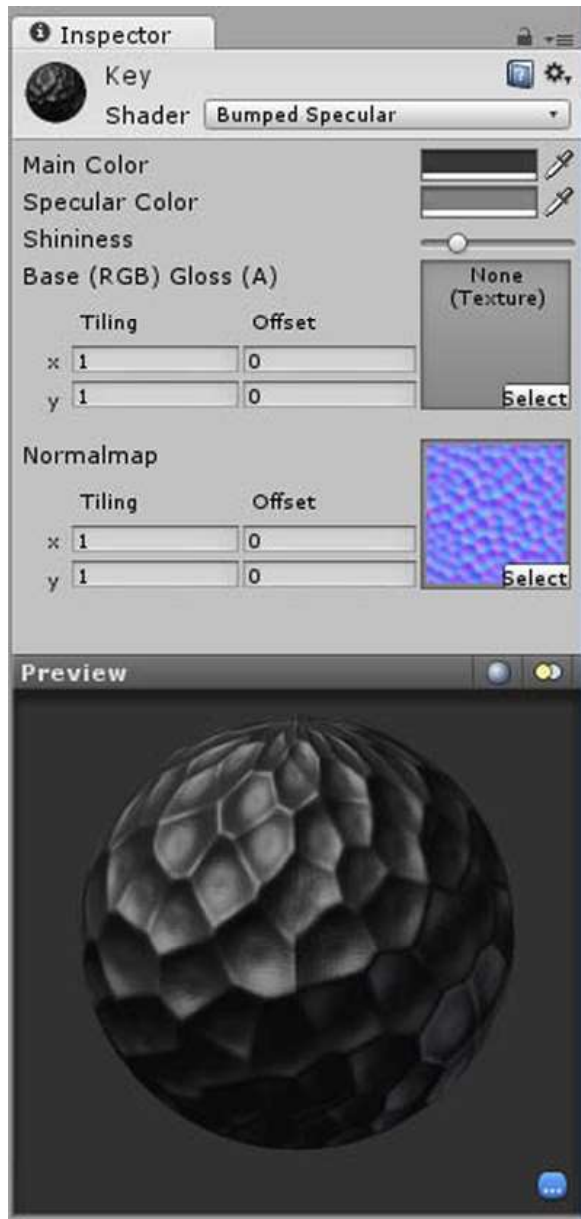


Figure 7-32. The HammeredBump material using the Bumped Specular shader

■ **Note** Normal maps are a means of creating virtual geometry for lighting calculations. When you start with actual geometry, the vertex normal for each vertex is recorded as a particular color on an image or texture. The texture map needs to be big enough to allocate at least 1 pixel for each vertex for full effect. In a game, the lighting calculations still take time to calculate, but the object itself has fewer vertices and takes up less overhead. As long as the normal map is not too big, the trade-off of polygons vs. texture memory is usually advantageous, providing the target platform has a graphics chip or card that can support it.

12. Next, select one of the keys in the Project view to check the end result in the Preview window, as in Figure 7-33.



Figure 7-33. One of the keys with the doctored material

The texture needs to be tiled more to fit the mesh better.

13. For the Normalmap, change both the x and the y Tiling to 5, as in Figure 7-34.

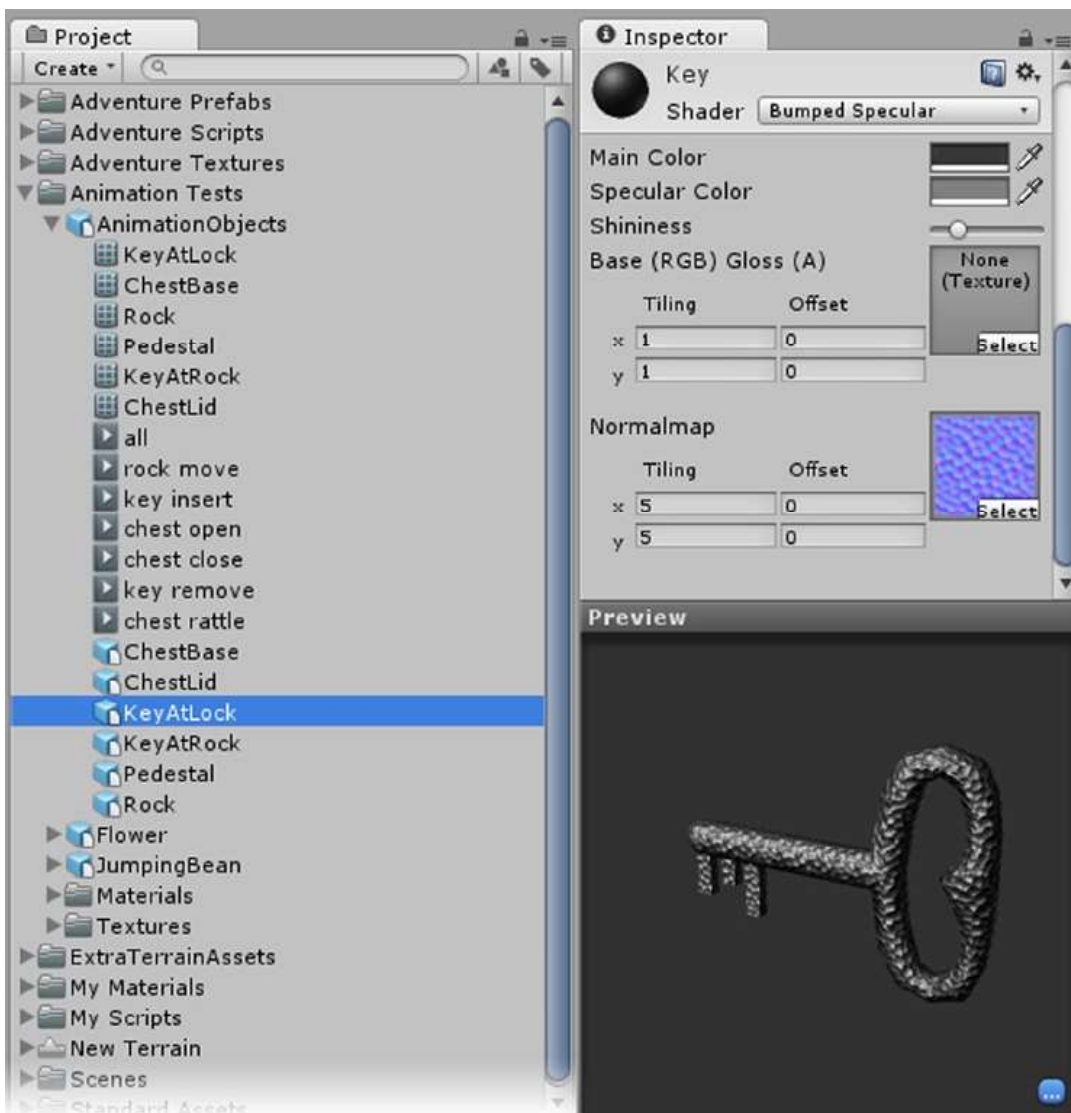


Figure 7-34. The tiled Normalmap

The material looks good in the preview, but you should check it in the scene.

14. Click Play and drive over to the key at the rock after it has been revealed (Figure 7-35).



Figure 7-35. One of the keys in the Game window at runtime

The key doesn't stand out much. You may want to darken the Main Color or reduce the Shininess. It's important to test art assets in the game environment before they are finalized. For the key, you may decide gold or brass would be a better choice.

15. Experiment with the key material's settings until you are happy with the effect.
16. Exit Play mode.

Changes made to materials during runtime are retained.

The Chest

The chest base and lid share a texture map, ChestFull. However, because the texture represents three different "materials" (wood, leather, and metal), you will also want to use a glossiness map along with a normal map, to differentiate each beyond the diffuse color of the texture.

With shaders in Unity, it is not unusual to put other types of 8-bit maps in the alpha channel. Each shader will show what, if anything, the alpha channel is being used for.

1. Select the ChestFull material.

Even when the object's material comes in with a texture, you'll note that the Main Color defaults to mid-gray. This color gets added to the texture map, allowing you to tint it on an individual basis. If your texture looks too dark in the scene, you can change the Main Color to white.

Inspection of the default Diffuse shader shows the Base texture using RGB.

2. Load the Bumped Diffuse shader and look at its parameters.

The Base is again RGB. The Normalmap is also RGB. If you are used to using grayscale bump maps, this will remind you that in Unity, Shaders use RGB Normal maps. Using a grayscale bump map instead of a normal map will produce odd results.

■ **Tip** Wait to convert your grayscale bump maps to normal maps in Unity, so you can easily adjust their bumpiness strength.

3. Select the next shader down, the Bumped Specular, for the ChestFull material.

This shader has a new parameter, Shininess, that will let you adjust the specular highlight of the material. More important, it uses the alpha channel of the Base texture as a specular mask.

4. Set the Shininess about halfway along the slider, to tighten the highlight.
5. Set the Specular Color to white.
6. Now drag in the Preview window so you can see the difference in the “materials” as the view rotates around the sample sphere.

The straps are somewhat glossy, but the metal parts are more so.

7. Select the ChestFull texture in the Project view.
8. In the Inspector, toggle the diffuse/alpha button at the top of the Preview window to see the alpha channel of the texture.

This alpha channel was designed to be used as a bump and shininess map rather than an alpha mask. You may notice that there’s a check box to Generate Alpha from Grayscale. That operation is permanent! If an alpha channel already exists (as with the ChestFull texture), it will be overwritten. If you want to experiment, make a duplicate of the texture first.

Figure 7-36 shows the superior contrast of the custom shininess map (center) compared with an automatically generated version. It will also be a better choice for generating a normal map.



Figure 7-36. The Diffuse texture (left), its custom alpha channel (middle), and an alpha channel generated in Unity, using its grayscale version (right)

9. Select the ChestFullBump texture.
10. Change its Texture Type to Normal map and click Apply (Figure 7-36).
11. Set the Bumpiness to about **0.05** and click Apply.

1024×1024 is a bit of overkill for one small chest.

12. Set its Max Size to **512** and click Apply.
13. Select the ChestFull material again.
14. Drag and drop the newly generated normal map into its Normalmap thumbnail.
15. Click Play and navigate around the chest.

You may decide that the 1024×1024 diffuse/shininess map can be reduced as well. Once an asset is in the game environment, you'll have a better idea of how big of a texture map it rates. In this case, the 512 version is slightly blurrier on the wood section, but still provides good definition for the straps and hobnails, so it would be a worthwhile conservation of texture memory. On a mobile platform such as iPhone or Android, you'll need to keep texture sizes fairly small, to stay within their total texture limits.

The Flower

Let's look at another material. The flower is a bit different from the other objects in that it has multiple materials on one of its objects.

1. Open the Flower object in the Project view.
2. Select the Stem prefab.

In the Inspector, you will see that there are two materials listed for it: leafstripey and leavesdual.

3. Select the Stem from the Flower in the Hierarchy view and open the Materials array to view its contents.

The array has two elements, the two materials, as shown in Figure 7-37.

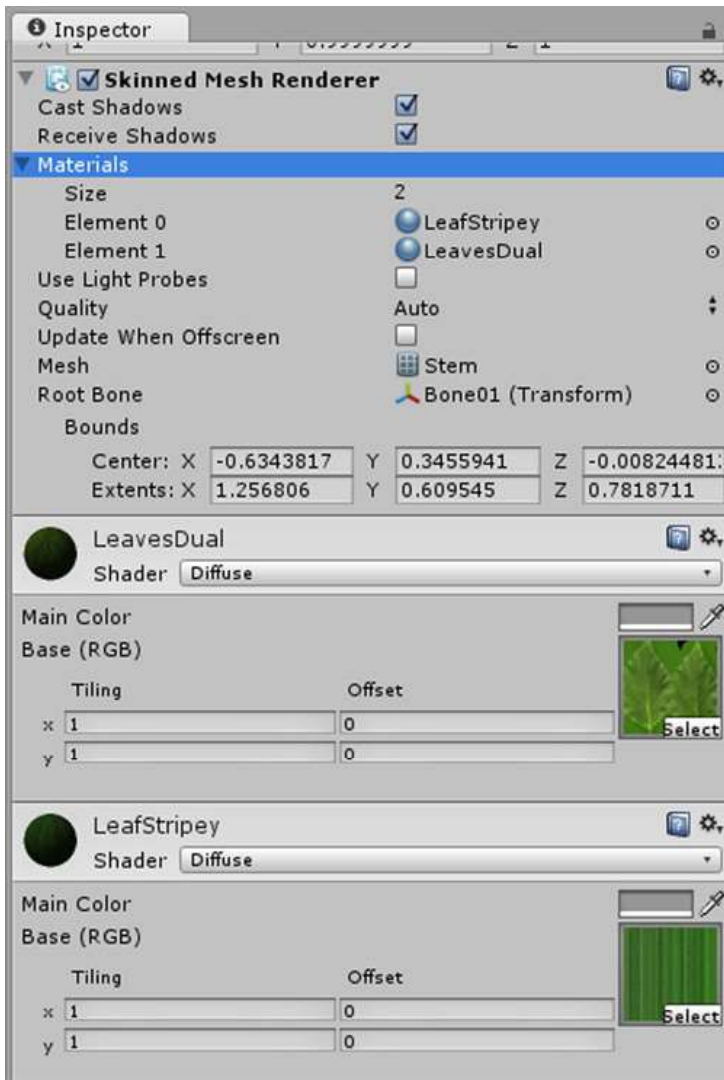


Figure 7-37. The Stem Materials array

■ **Tip** Arrays always start counting their elements at 0. An array of 2 has element 0 and element 1.

If you wanted the flower to be an action object, that is, one you'd need to click, it would be better to use a single unwrap map to make the textures for the leaves and petals. Because you are only going to trigger its animation from a different object for now, you don't have to worry about it.

■ **Tip** Assets with unwrap textures are generally easier to handle in games, but the trade-off is the possibility of using more texture memory to get the same resolution or settling for a reduction in quality.

The pollen ball in the center of the flower presents some nice texture possibilities. Because the pollen object is deep in the bone hierarchy, you will have to expand the Flower object to gain access to it.

Remember to hold down the Alt key when you open the group in the Hierarchy view.

4. Locate the pollen ball at the bottom of the Bone07 hierarchy and select it.

It is using a material called PollenBall.

5. Change the shader to Vertex Lit, set the Main and Emissive colors to mid-gray, the Specular Color to red, and the Shininess to the far left.
6. Drag the Pollen texture from the Textures folder onto the Base texture thumbnail.
7. Change the Tiling to 3×3.
8. Click Play to see the results when the flower opens up (see Figure 7-38).



Figure 7-38. The pollen ball's new shader

9. Try adjusting the Main Color down to black to see the full effect of the red specular.

Note that the changes to shaders are not lost when you stop Play mode.

The Rock uses the GrndRed texture.

1. Change the GrndRed's shader to Bumped Diffuse.
2. Use Ctrl+D to duplicate the Asphalt texture and turn the clone into a normal map.
3. Add the new normal map to the GrndRed shader.

Shadows

Now that you've used the materials to make the imported objects look much better, you are probably thinking some shadows would help to make them look like they belong in the scene and aren't just "Photoshopped" in. In Unity Pro, you can set the main light to cast soft shadows, and everything will look much nicer—at the cost of some frame rate, of course. If you are using Pro, you may already be using the soft shadows, and your scene may more closely resemble the screen shots.

Let's address shadows in Pro first, as it is more a matter of tweaking the shadow strength. Depending on where your light source is, you may need to adjust the shadow parameters.

1. If you are using Unity Pro, open the Lightmapping editor; change the Mode to Dual Lightmaps and Bake.
2. Select the Directional light.

In the Inspector, you can see the shadow parameters. Strength controls how dark or dense the shadow is.

3. Adjust the Strength until the dynamic shadows match the baked shadows.

Take a look at Figure 7-39. On the left, you see dual lightmaps with Lightmapping set to Auto for the directional light. In the Project's Quality settings, the Shadow Distance was set to 150. The real-time shadows blend into the lightmap shadows 45 meters out from the Main Camera. On the right, I turned off the Use Lightmaps parameter (open the Lightmapper and uncheck the option in the Scene view), to show where the real-time shadows blend with the lightmapped shadows.

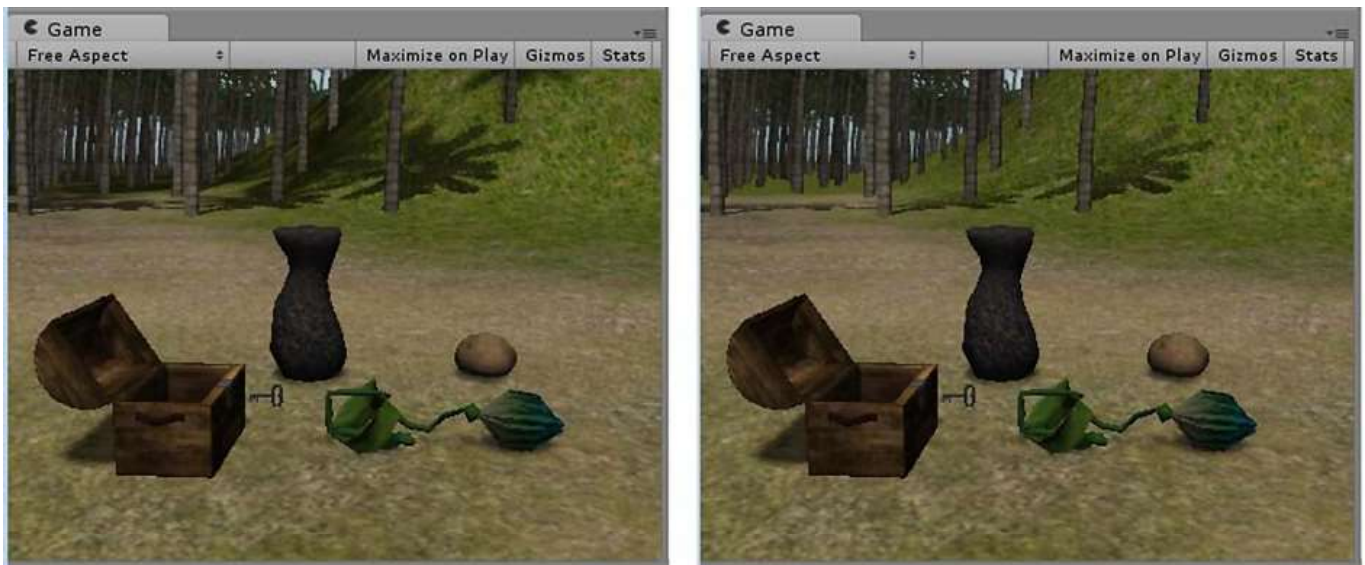


Figure 7-39. Shadows in Unity Pro (left), showing the end of the dynamic shadow zone (right)

If you are using Unity Pro and have shadows, you may want to investigate Shadows further in the reference manual. You'll find some very nice examples of "Cascades" and see how they can improve the look of your shadows. You can find the Cascades setting in the Quality Settings.

Static Objects and Shadows

If you don't have Unity Pro, you have a couple of choices: no shadows or fake shadows. If the objects never move, they can be marked as Static, and shadows will be baked for them. But for animating objects, leaving a shadow behind as the object animates away from its original position is going to look pretty silly. In our example, some objects, such as the flower and chest lid, are probably fine with no shadows. Others, such as the rock, will need some help. Shadows, in any form, will cost. If they are baked in, they cost texture memory. If they are dynamic, they cost processor time. If we fake a few dynamic shadows with Unity's Blob Shadow preset—a light with a negative strength—they also cost processor time, but not as much.

Using shadows increases the quality of a scene, removes visual ambiguity, and, along with specular highlights, helps define a 3D shape. To put it bluntly, things just look better with shadows, but dealing with shadows in real time is vastly different than in DCC programs.

To generate baked secondary shadows in Unity, the object must be marked as Static.

1. Select the ChestBase in the Hierarchy panel.
2. In the Inspector, at the top right, check Static (Figure 7-40).

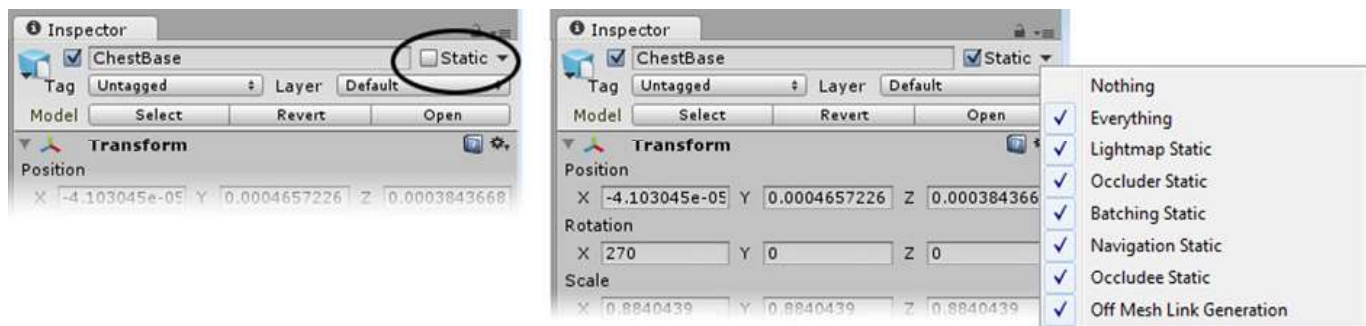


Figure 7-40. Selecting the Static parameter in the Inspector, and the drop-down showing the default results

The Pedestal does not animate either.

3. Set the Pedestal to Static as well.

The next time you bake out the terrain lighting, the shadows will be added for the static objects.

Blob shadows are often used for generic character shadows when you're creating games for low-end platforms. Objects such as the rock are good candidates for a blob shadow.

If you don't see a Projectors folder in the Standard Assets folder (Figure 7-41), you'll need to import that package.

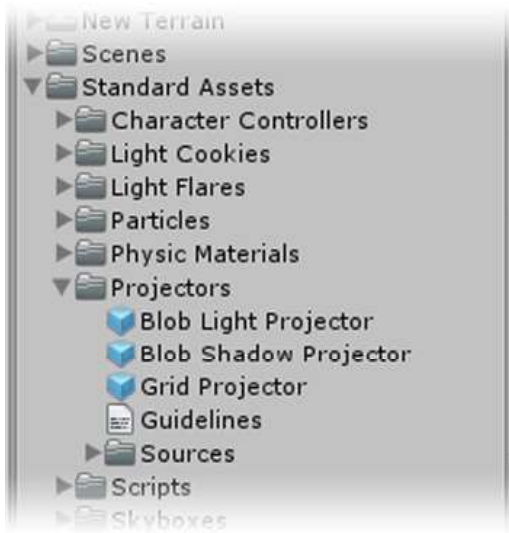


Figure 7-41. *The Projectors folder*

4. From the Assets menu, select Import Package ► Projectors.
5. Select all and Import.
6. If you are using Pro, set Shadow Type on the scene's Directional light to No Shadows.
7. From the Projectors folder, drag a Blob Shadow Projector into the Scene view.
8. Position it over the RockGroup and move it up or down until the "shadow" looks about the right size (Figure 7-42).

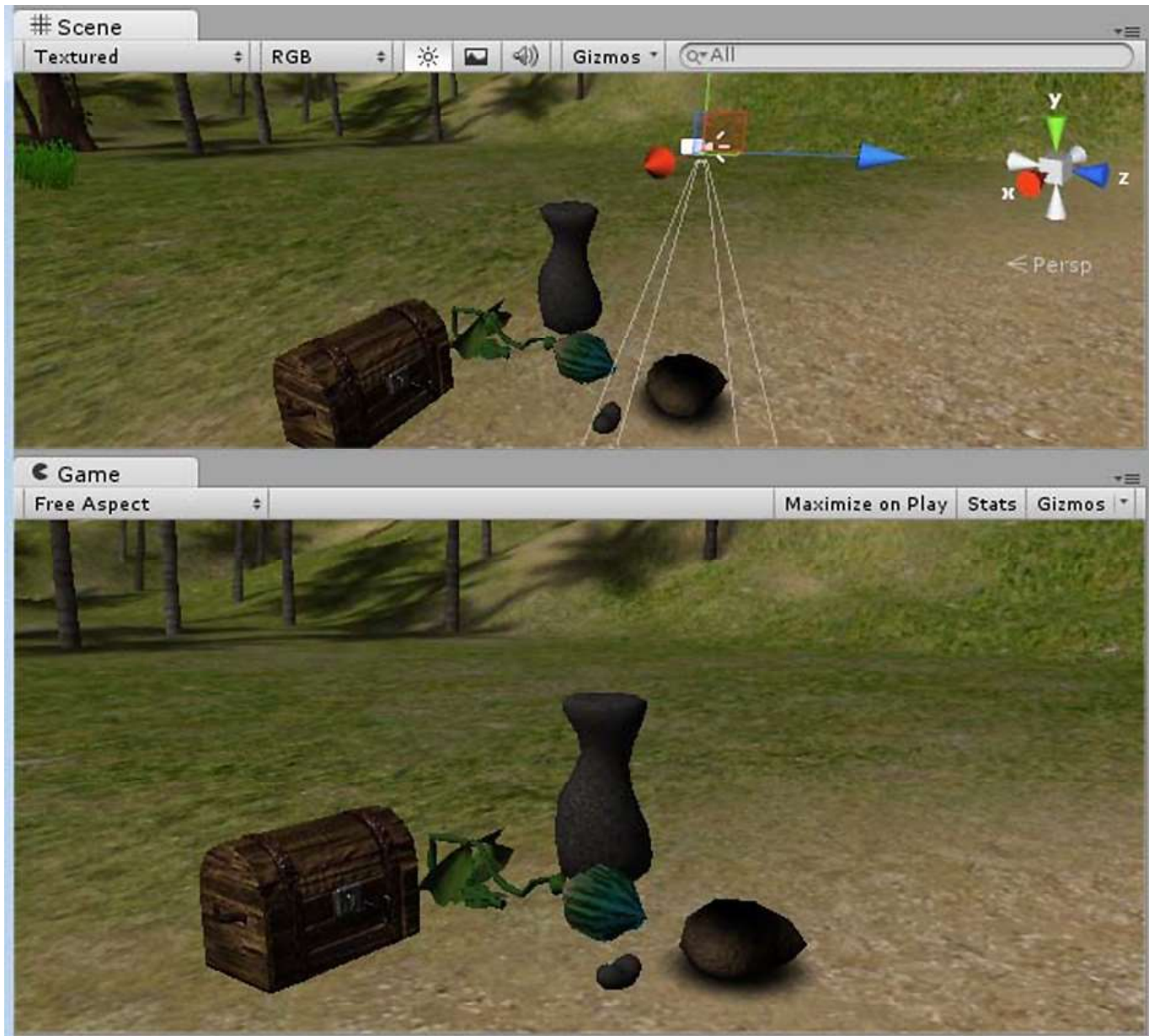


Figure 7-42. Positioning the blob shadow

Right now, the projector light projects on the top of the rock group. With a character, you'd use layers to prevent the blob shadow from projecting on the character itself. You can use the same technique on the rock or any of the action objects. For the test subject, let's adjust the clipping plane until the shadow looks correct and then create a Layer, so we can exclude the rock itself from receiving the shadow. Layers provide a way to include and exclude objects and effects from mainly lights and cameras. The Blob Shadow is actually a kind of light with a negative value.

1. In the Inspector, change the Blob Shadow Projector's Projector component's Aspect Ratio to 1.4, to elongate it slightly.
2. Rotate the Blob Shadow Projector until it matches up to the rock better.
3. Change the Far Clip Plane to about 15—the rock won't be lifting very high.

Before you can use the Ignore Layers parameter, you will have to set up a layer for it to ignore. You will be doing more with layers later, so consider this a sneak peek. Layers and tags are a means of including or excluding objects. Layers are generally used with lights, and the Blob Shadow is a form of light.

4. At the top of the Inspector, click the Layers drop-down and select Add or Edit Layer (Figure 7-43).

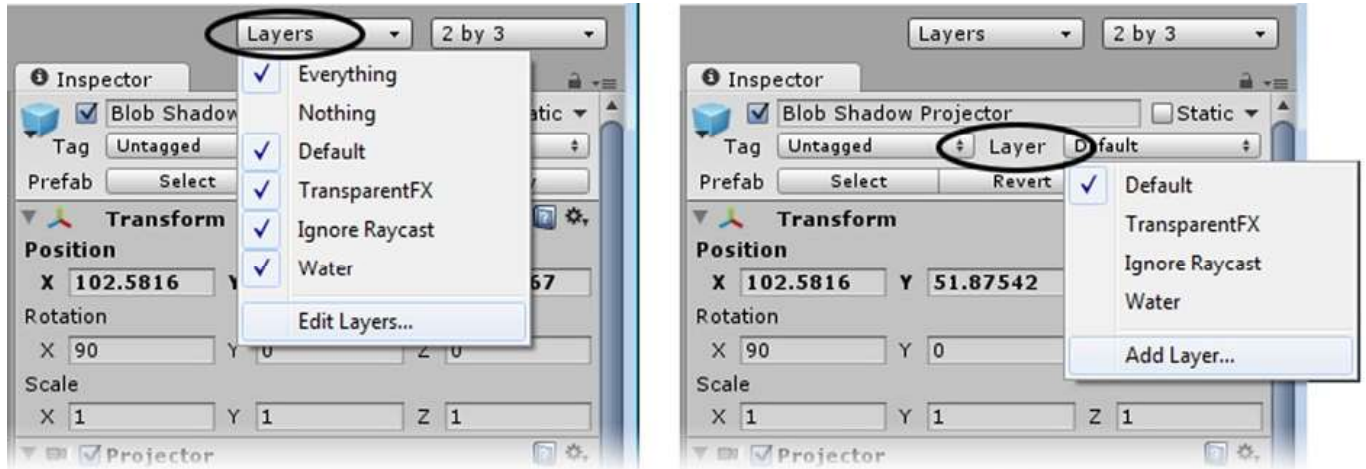


Figure 7-43. Accessing Layers

5. To the right of User Layer 8, click to locate and activate the text field and name the layer NoBlob.
6. Select the Blob Shadow Projector again in the Hierarchy layer.
7. Click the drop-down for Ignore Layers and select the new NoBlob layer.
8. Select the Rock Group.
9. At the top of the Inspector, click the Layer drop-down and select the NoBlob layer for the Rock Group and agree to Change Children.

The rock is now ignored by the Blob Shadow Projector, so only the ground receives the fake shadow, as shown in Figure 7-44 (left).



Figure 7-44. Adjusting the Projector settings to fix the shadow(left), and the Rock Group in action (right)

10. Adjust the Blob Shadows parameters until it looks correct.
11. When it looks about right, drag the Blob Shadow Projector onto the Rock in the Hierarchy view.
12. Click Play and watch the Rock and its “shadow” (see preceding Figure 7-44 [right]).

Because of the falloff built into the shadow via the Gradient map, the “shadow” gets larger and paler as the rock, the projector’s parent, moves up away from the ground.

13. Stop Play mode.
14. Deactivate the Rock Group.
15. If you are using Unity Pro, go ahead and turn shadows back on for the Directional Light.
16. Save the scene and save the project.

Summary

In this chapter, you imported assets for your scene and learned how to process them for animation and materials.

When importing animated assets, you learned how to correct scale. When the animated objects come in together, their animation component, either the newer Mecanim Animator or legacy Animation, was on the parent. Single animated objects, you found, need to be parented before you adjust their placement in a scene. You also learned how to split the Take 001 animation clip into individual behaviors.

On importing objects, you found that, as a default, Unity creates generic materials using the original material’s Diffuse texture map as the name. If no texture map was used for the material, the Unity material was named for the file and appended with the original material name, unless a different option was chosen.

After importing, you discovered you could use different shaders to improve the look of the materials. One of the most important concepts was that of using the alpha channel component of a texture as something other than an alpha channel mask. You also found that you could have Unity generate normal maps, but that you needed to duplicate the original texture before doing so. Additionally, you found that Unity converts all texture maps to dds format internally.