the Update() function in that it executes for an object every game loop. However, with `LateUpDate()` it occurs as the very last function called for an object so that it can take into consideration any related processing occurring in the same game loop.

Step 10: Open the project with the warehouse. Create a new C# file and enter the same code from Listing 2.16. Attach this script to polySurface437. This is the surface of the conveyor belt in the room where you placed the curtain. Play and watch the conveyor move!

2.7.3.2 Blob Shadows

Shadows give a scene an extra dimension of depth and add to visual realism. Generating shadows is processor intensive. Although shadows in game environments are covered in Chapter 7, a quick and easy method for generating processor light shadows called Blob Shadows is introduced here.

Usually the game rendering system calculates shadows based on the position and intensity of lights and the position of game objects. When many real-time shadows need to be calculated, such as those of moving objects like characters, it can slow the frame rate considerably. This is a big problem for games on mobile devices where such shadowing is not practical.

Unity Hands On

Blob Shadows

Step 1: Download *Chapter Two/BlobShadows.zip* from the website, unzip, and open in Unity. In the Project, double-click on *blobshadowexamples* in the *Scenes* folder to open the scene. The Scene will appear with a character standing on a plane.

Step 2: Select Assets > Import Package and import the Effects Package from the book's resources or download and import from the Asset Store.

Step 3: Locate *Blob Shadow Projector* in the Effect > Projectors > Prefabs folder and drag and drop it onto the Male game object. A frustum object will appear under the Male model. This is the Blob Shadow Projector.

Step 4: Move the projector up until it is just above the top of the model's head. A black blob will appear on the ground. This is the blob shadow. The shadow is created using a material with the texture of the round black blob on it. When the projector intersects a plane, as it does here with the ground, the black texture is drawn inside the intersection area of the plane and the projector frustum.

Step 5: In its current state, the Male model is also inside the projector frustum and therefore the shadow is drawn on it too. To fix this, we place the model into another drawing layer. Select the Male object in the Hierarchy. In the Inspector at the very top to the right of Tag is a property called Layer. Click on the drop-down box next to Layer and select *Add Layer.*

Step 6: In the Tag Manager that opens in the Inspector, next to *User Layer 8*, type in *character*. This will create a new drawing layer called *character*.
Step 7: Select the Male object in the Hierarchy again. In the Inspector, set its Layer to character using the drop-down box.
Step 8: Select the Blob Shadow Projector in the Hierarchy. In the Inspector's Projector Component, set the *Ignore Layers* property to *character*. The projector will now ignore anything in the character layer, which in this case is the Male model, and draw the shadow object everywhere else it intersects. This is a very effective way to add convincing shadows to objects without adding too much processing overhead.

2.7.4 Billboards

Billboarding is a technique that uses planes to fake a lot of background scenery. A billboard is a plane usually having a partially transparent texture applied to give it the appearance of being a shape other than a square. Common uses for billboards are grass, clouds, and distant trees.

To give the illusion that the billboard is viewable from all angles, the plane orientates itself constantly so that it is always facing the player.

Unity Hands On

Billboards

Step 1: Download the file *Chapter Two/Billboards.zip*. Unzip and open the Unity scene called grass. In it you will find a quad with a grass texture and a ground plane. The grass texture has transparency that allows you to see through the areas where there is no part of the grass image.
Step 2: Import the Character Controller asset package and drop a First Person Controller from it into the scene so you can walk around on Play.
Step 3: Create a new C# file called FaceCamera and attach it to the grass quad. Enter the code in Listing 2.17.

Listing 2.17 Script to create a billboard from a quad

```
//FaceCamera.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FaceCamera : MonoBehaviour {

    // Use this for initialization
    void Start () {
    }
```

```
    // Update is called once per frame
    void Update () {
        //note the minus in here as a Unity Quad has its Z
            axis facing in the opposite
        //direction to its textured side.
        this.transform.LookAt(-Camera.main.transform.
            position);
    }
}
```

Save and Play. The billboard will have disappeared. Why?

If you take a look in the Scene, you will notice the quad is still there, but the grass is facing away from the player. The LookAt() function orients an object such that the blue forward-axis (*z*) faces the object in question. In this case, the image just happens to be on the opposite side as shown in Figure 2.51. This means we need to not only look at the camera with the quad but also rotate it by a full 180° around its up axis. To do this add:

```
this.transform. Rotate(new Vector3(0,1,0),180);
```

as the last line inside the Update() function.

Now when you play, the grass will always face the player.

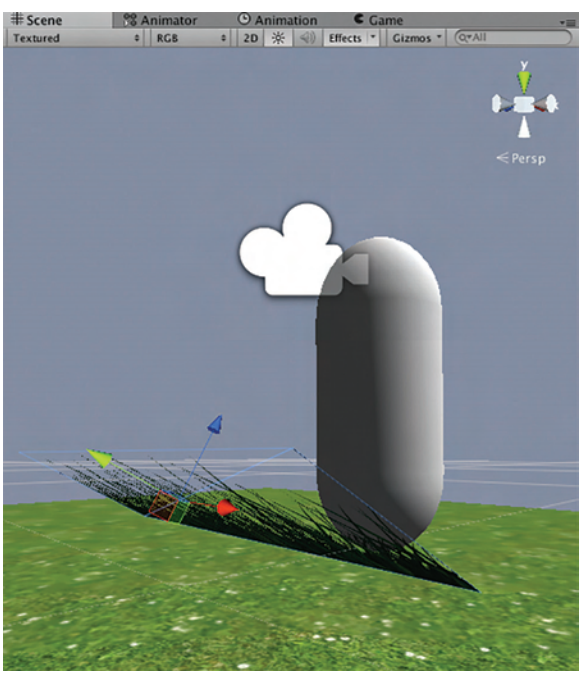

FIG 2.51 A billboard with the texture on the wrong side.

Step 4: To see the full effect, duplicate the quad, and place it around the scene multiple times. Play and walk around.
Step 5: When you walk over the top of the grass it will lay down flat. If the billboard were a tree you would not want this to happen. Rather, rotating around the *x* axis should be turned off. That way the object will stay vertical and only turn around its *y* axis. To allow for this, modify FaceCamera to the code in Listing 2.18.

Listing 2.18 Billboarding script that allows the *x* rotation to be turned off

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FaceCamera : MonoBehaviour {

    public bool stayUpright = true;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        this.transform.LookAt(-Camera.main.transform.
            position);
        if(stayUpright)
        this.transform.eulerAngles = new Vector3(0,
            this.transform.eulerAngles.y,
            this.transform.eulerAngles.z);
    }
}
```

Step 6: The grass will not bend over as the FPC approaches it. You can now turn this feature on and off using the tick box for StayUpright in the Inspector when FaceCamera.cs is attached to a game object.

More often than not, billboards are used on horizon lines and in the distance. Because they do not stand up under close scrutiny, you may want to use them en masse, but in areas of the game environment the player cannot quite reach.