Chapter 5

Using a 3D environment

This is an optional chapter about using the 3D area to make a little nicer set-up. I feel like knowing a little more makes the programming part easier to understand. And Unity3D uses most of the same concepts as any other 3D non-game program, so not a waste to learn even if you never plan to use Unity. And sometimes playing with the system can give you the idea for more mini-programming projects later.

But you can skip this section. Much later, when we bounce things around, We'll need some of this (a floor and walls to bounce from.) But not for a while.

5.1 Making a small room

5.1.1 Intro

Like many 3D programs, Unity starts with a completely empty game world – like outer space. The horizon you see is fake – there really isn't even a floor or ground. I think it's helpful to make a little room for our Cube – a floor and three walls, so maybe more like a diorama.

Doing that will involve learning just a little about how to use the controls, and what some of the numbers mean.

I'm going to explain the steps, but if you get lost or just feel like it, the Internet has a lot about simple Unity3D controls. I won't be offended if you stop and look something up.

5.1.2 A floor

We can make a floor using a stretched-out Cube:

Make another Cube (`GameObject->3DObject->Cube` again.) You might want to rename it "floor," to avoid confusion later (the usual way: click to select it, then click again.) To put it where a floor should be, go to its Inspector

position slot and enter (0,-5,0). You don't need to, but the TAB key jumps to the next – you can enter 0 for X, then tab to Y.

If you look at it in the big Game window, it should be lower-centered, but still just a small cube. To stretch it out, find Scale in the Inspector (two spaces below position.) A clever way is to move the mouse left of the X box (over the letter X.) It should turn into a double-ended arrow. That's an invisible slider for the x-scale. Left-click-drag will let us stretch the Cube wider. The same thing with Z will make it deeper, and more floor-like. Adjusting those can give it a nice floor-like look.

The Esc key will cancel a drag-slide (while you're holding the button down,) or the usual ctrl-Z will undo it. Or you can just type in numbers for X and Z, like we did for position. Drag-slide is just a nice short-cut.

The camera probably cuts off a lot of our floor. For fun, we can take a look at the whole thing using **Scene** view. The biggest window has tabs **Game** and **Scene**. Game is locked to what the game camera sees, while Scene lets us move around.

To move around: select the **Scene** tab. You should see a view of the floor from a different angle. Then double-click the floor in Hierarchy – the Scene window will zoom-center on the floor. For fun, you can double-click the Light or Camera or the other Cube and watch it zoom there.

If you select MainCamera, the Scene window will show a little white pyramid coming out of it – four lines shooting out from it at angles. Those show you the camera view – that's what we've been seeing while it runs, and what the Game tab always shows.

If you move the mouse over the Scene window, the scroll wheel moves you in/out. Right-click-drag spins around what you're aimed at (called orbiting.) The arrow keys will slide left/right/up/down (the Scene window has to be active. You may have to click on it.) Clicking the green/blue/red gizmo on the upper-right snaps to that angle.

Don't worry about getting lost. That's what double-clicking is for. It takes a while to get used to moving around in a 3D program. Scene's only purpose is to let you walk-around and adjust things, so don't worry about breaking it or putting it back the way it was.

A fun trick with Scene view is you can use it while running the program. If the Cube goes off the edge, you can switch to the Scene tab, scroll-zoom out, and watch it keep going. You can even double-click the moving Cube to follow it.

5.1.3 walls

We can make the walls the same way as the floor: make another Cube, move it where the wall should go, and scale it.

The moving cube is going from x= -7 to 7, so that's where the side walls should go. We can make a Cube, then hand-enter (-7,0,0) for Position, then slide on Scale Y and Z to grow it tall and deep.

Doing that in Scene view will let us get a good angle on it. But doing it in Game view will let us see the real way it will look. I usually go back and forth.

For the wall on the right, we can use Copy/Paste on the first wall, and change position-X from -7 to 7.

The back wall is just for looks, so not as important, but fun to make. We can start its Cube at maybe `(0,0,-6)`. The idea is, we know our moving Cube goes through 000. So z at -6 will put the back wall 6 units behind that, which seems far enough. Then we can slide-stretch X and Y to make it wall-sized.

For a test, enter (-7,0,0) for the position of the moving Cube. That might move it partway *inside* the left wall. Gah. Maybe that's fine (the computer won't care about one thing inside another.) Or maybe the walls should move a little further apart.

Just so you know, the Cube is 1x1x1 and centered. When its at -7, it's left edge is at -7.5.

After, if you didn't do it already, go to Scene view and look around.

Notes:

- I put the floor at -5. Why not 0? The system doesn't have any special meaning for particular numbers, so a floor can be anywhere. The moving Cube we have now stays at a height of 0, so -5 seems like a good place for a floor.
- The size of the window can also be anything. We could position the camera and resize it so x goes from exactly 0 to 100 across the screen. But that's a pain: the camera starts at -7 to 7, and those numbers are as good as any others.
- The axis also depend on where the camera is aimed. Our camera is looking from the front, so x is sideways and y is up. A topdown camera, looking down at the floor, would still have x going sideways, but z would be up and down.
- In the upper-right corner of Scene view, you should see a little XYZ gadget, showing the view angle. The three colored ends point in the positive direction of that axis. You can also click on any of those cones, and it should snap views (it stays centered on the item you double-clicked, so it's kind of neat.)

5.2 Color, Textures

Coloring the shapes seems frivolous. But in my test scenes I've found that colored stuff really helps me see what's happening. Plus it looks really neat, and might give you some ideas for things to program.

The standard way color and pictures are placed on 3D objects is using one extra thing, called a **Material**. If you want a red cube, you make a Material, color it red, and place that Material on the cube.

Having to make a Material first seems like a pain. One reason for it is, you can set the color once, then use that Material on all the objects you want. The other is there are also lots of settings, and a Material is a good place to store them. For example a picture of a cat can be doubled in one Material, set to a green tint in a second, and normal in another.

Here are the steps to turn the floor red:

- In the **Project** panel select `Create->Material`.
- Enter a name for it. You should have a little ball icon with "New Material" written, waiting for you to enter a better name. You can easily rename it later, so anything is fine.
- The Material should appear in the Inspector. If it doesn't, select it again. The top item should say **Albedo**. That's a fancy word for "the basic color." Click the color picker and find one you like.
  If you've never used one before, the *top* bar is the color. The long right-side bar and the big middle square are used to adjust it.
- Drag the material into the Scene. As it goes over each object, you should see it change color. Drop it on the one you like.
  Just so you know, you can also drag the Material to the name, in the Hierarchy panel. The real spot, if you need to check later, is in the Inspector for the item inside the `MeshRenderer` section. Look where it says `Materials`. Pop that open if needed, and look in `Element 0`. That's the real place Materials go (don't worry about element 1 or 2. Only specially made objects ever have more than one.)

Using a few Materials we can now turn the ball yellow, or make the back wall blue. The ball should stand out more. Plus it's fun.

Some notes: notice how the color-picker, in the lower-right, shows red, green blue numbers as 0-255. That's the other standard color range. From last chapter we know Unity prefers 0-1. The color picker secretly converts the ranges.

The line changing color should make a little more sense now. The one: `GetComponent<Renderer>().material.color=`. The color is stored in a material (we still don't know what a Renderer or a Component is, but 2 out of 4 is

pretty good.)

A fun thing is to add a tiled picture (we call pictures *textures*) to the floor. My floor is 14 wide and 7 deep, so, I'll get a picture to repeat that much. The steps are:

- Find any picture in your computer. Try to get one that you can tell when it repeats (so, not a solid color.) Drag it into the Project panel in Unity. It will be copied.
- Have one Material already created and on the floor.
- Select that Material and go to Albedo again. We're going to put the picture here. One way is to drag the picture into the square. Another is to click on the small circle in a circle. That's the "show me my choices" icon. It should open a window that lets you pick an image.
- To make it tile, still on the Material, look down for *Tiling*. There are two – make sure it's the first one, under "Main Maps." Enter 14 for X and 7 for Y. You should see the image repeat that many times.

This can be nice for measuring. If your object is exactly 12 wide, and a picture tiles 12 across, then each picture is exactly one unit.

One thing to watch for, you can reuse a material for several Cubes and they *share* it. If your floor material is also on something else, it will also get the picture. If that's a problem you can use Edit->Copy on a Material.

5.3 Adjusting the Camera

For fun, we can move and re-aim the camera and change the "perspectiveness" of it. Sometimes a better view can make testing easier.

Suppose we want to move the Cube from X=0 to 10, which means we need to adjust the camera to see that range. As a guide we could change the floor to run 0 to 10: make it 10 units wide (just type a 10 into the x-scale) and change the X in position to 5.

Or, an even better guide, we could create and place dummy cubes on each side: x=0 and another at x=10. Together with the new floor, that should give a pretty good indicator for when the camera is aimed correctly.

Obviously the camera should be at x=5. Since it uses perspective we can slide the camera in and out (using it's Z position) until the edges are correct.

We can change the edges of the camera another way. The amount of perspective is controlled by FieldOfView, in the Camera section of the Camera's Inspector. It's starts at 60, for no reason. We can slide that around to adjust the edges. If you want a mostly flat perspective, pull the camera way back and

narrow FieldOfView.

Another fun thing is to angle the camera down a little. You can click-slide any of the camera's rotations. Y pans (looks sideways,) Z cocks it, and X angles up/down.

You can eventually get a nice downward view by going between angling x-rotation down a bit and sliding up the y-position. Putting the Cube at 000, and those dummy cubes on the sides, can help you get it right.

Again, none of this really matters. It's just nice knowing where x= -7 to 7 came from, or why changing z position just changes size a little. And sometimes it helps with testing to be able to make some landmarks, like walls or things at some exact position.