50.017 Graphics and Visualization

Unity Lab Session 1: Introduction to Unity with the Roll-a-Ball project

(Originally created by Unity Technologies and documented by Tian Feng)

1 Overview

In this lab session you are about to have an overview on Unity and learn how to develop a small and simple game called Roll-a-Ball by Unity.

2 Quick Questions on Unity

a) What is Unity?

Unity is a game development ecosystem: a powerful rendering engine fully integrated with a complete set of intuitive tools and rapid workflows to create interactive 3D and 2D content; easy multiplatform publishing; thousands of quality, ready-made assets in the Asset Store and a knowledge-sharing community.

It is necessary to distinguish Unity from other modeling applications (e.g., 3DS MAX). As a game engine, Unity concentrates on how to organize existing models for making a story. When developing with Unity, developers follow a style called *Component-based Programming*.

b) What is Component-based Programming?

In Unity each object, no matter a camera, a light, a text or a model, is called a *GameObject*. And all properties and functionalities of a GameObject are represented as *Components*, that is, a GameObject contains a number of Components.

Unity encapsulates popular properties (e.g., translate and material) and functionalities (e.g., physics and collision detection) into Components. Just add what is wanted when needed. Besides the encapsulated Components, Unity allows developers to add scripts as Components to control GameObjects. So far, JavaScript, C# and Boo are supported as scripting languages. We use *C#* during our labs.

c) What is C#?

C# is the major programming language in Microsoft .NET Framework. As C# is very similar to Java in grammar and syntax, any Java developer should be familiar with C# in a very short time.

Below is an official C# tutorial which may help to learning the language.
https://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx
http://unity3d.com/learn/tutorials/modules/intermediate/scripting

d) What is the world coordinate system in Unity?

The world coordinate system in Unity is the same to that in OpenGL, which is a Right-handed Cartesian Coordinate. Generally speaking, when we put the camera at (0,0,1)

and let it face to (0,0,0), the X axis is the one from left to right, the Y axis is the one from lower place to upper place and the Z axis is the one from further place to camera.

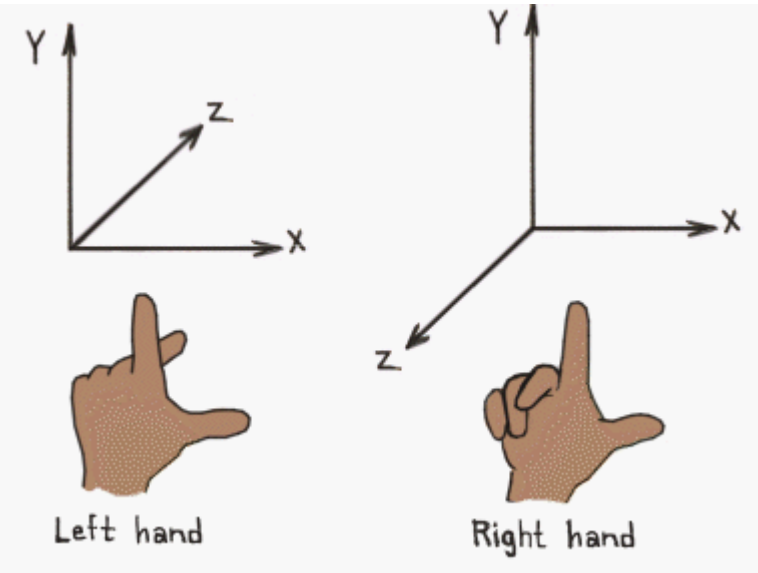


Figure 1. Cartesian coordinate systems.

e) How is the Unity interface like?

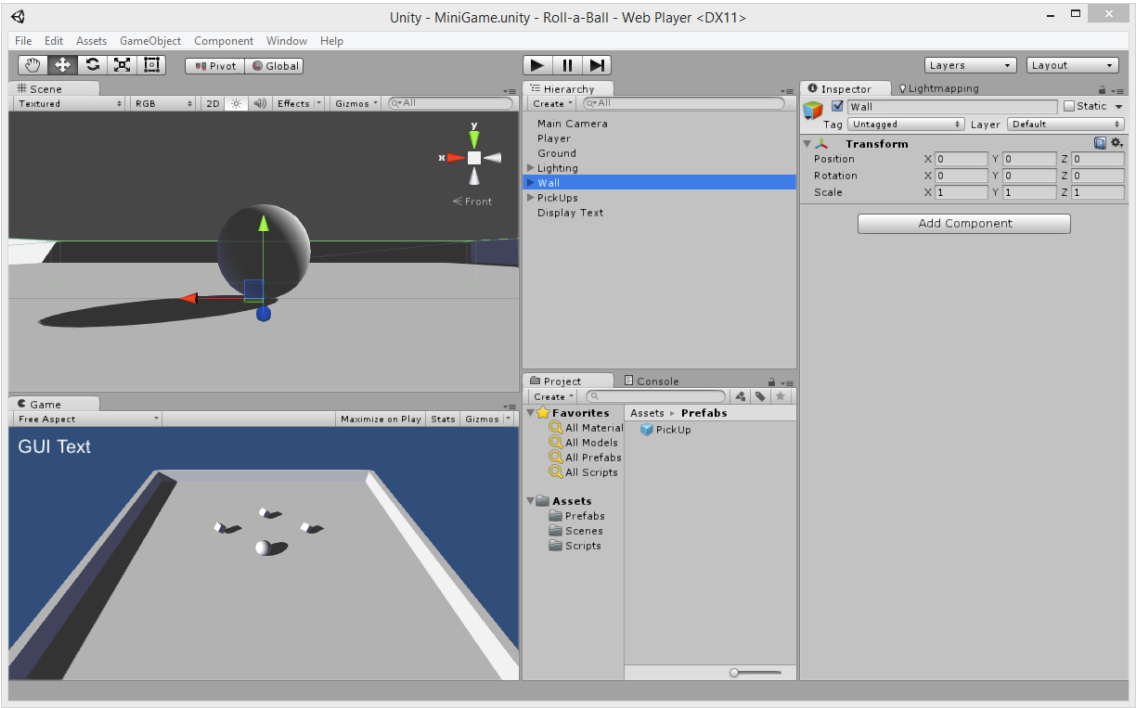


Figure 2. The interface of Unity.

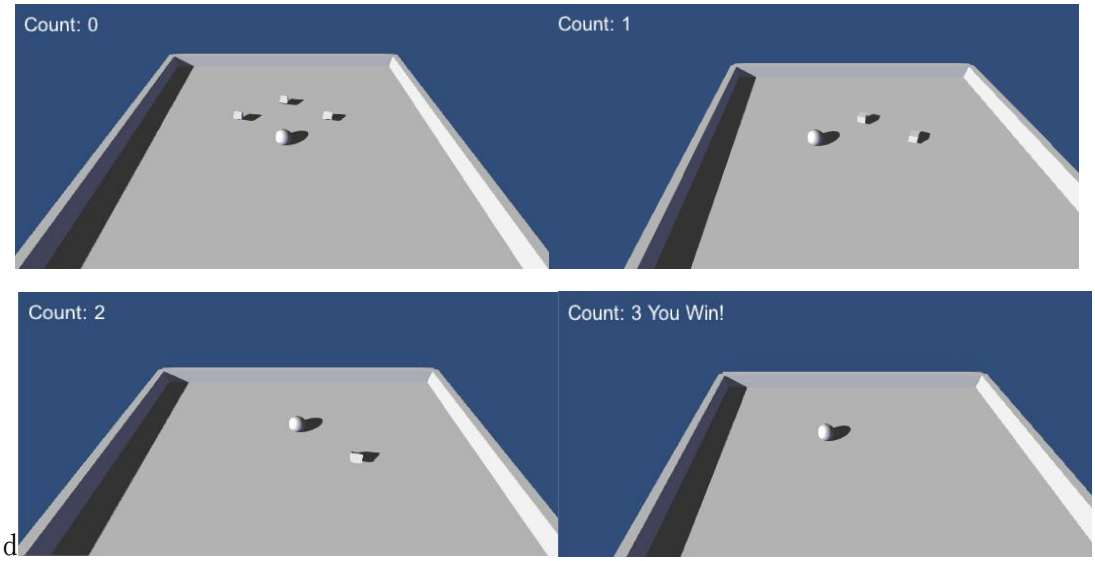
- **Scene** (upper left): You can observe all your GameObjects freely in this window.
- **Game** (lower left): You can observe what the final output will be like in this window. When testing the project, all GameObjects will be updated in Scene and Game windows, but in different viewpoints.
- **Hierarchy** (upper middle): You can view/add/delete/order/structurize all your GameObjects in this window. Noted that each GameObject is a “logic” object

which should only exist in the memory of your computer.

- **Project** (lower middle): You can view/add/delete/order/structurize all your assets in this window. Noted that each asset is a “physical” object (or a file) which should exist in the storage of your computer. A GameObject can be stored as an asset for easy utilization.
- **Inspector** (right): You can view/add/delete/update all Components of a GameObject in this window. Note that each GameObject (even an empty GameObject) contains at least one Component called *Transform* which is in charge of its position, rotation and scale information.

3 Project: Roll-a-Ball

a) Introduction



d

Figure 3. Captures of the output game.

We are about to develop a small game named Roll-a-Ball. The user is able to control the ball, by keyboard for hitting the three rotating cube pick-ups. When hit by the ball, the pick-up vanishes. The text located at the upper left corner of the screen shows the number of hit pick-ups. When all pick-ups are gone, the text displays “You Win!”

b) Setting up the game

- **Creating new project**: File->New Project. Created a new project named *Roll-a-Ball*.
- **Save scene**: File-> Save Scene. Save the initial empty scene and name it *MiniGame*.
- **Create the playground**: A playground is necessary for the ball to roll around. As simple geometries (e.g., plane, cube and sphere) can be directly used, we may intuitively construct the playground with a plane and four cubes. By the Translate Component, we are able to adjust the shape of geometries and also locate them to right places.
- **Create the ball**: As mentioned, we create a sphere as the ball and locate it at the

center of the playground. Noted to set right Y value to locate the ball on the playground so that it is able to move there. Besides, we need to add the RigidBody Component to the ball as to make it capable with physical properties.

c) Moving the ball
   - **Add script as Component to control the ball**: In order to roll the ball by input (e.g. keyboard), we add a C# script to the ball as Component. By Component->Add from toolbar or AddComponent->New Script->CreateAndAdd from Inspector, we create a C# script file named PlayerController. In this script, we use the class Input and Translate to read the movement and update the position of the ball.

d) Moving the camera
   - **Add script as Component to control the camera**: In order to let the camera move with the ball, we add a C# script named CameraController. In this script, we use the class Translate to update the position of the camera by adding a given offset.

e) Creating pick-up objects
   - **Create pick-ups**: We create three cubes as pick-ups and randomly locate them on the playground. Noted to set proper Y value to locate pick-ups over the playground and rotation value to make them look interesting. Instead by separately creating pick-ups, we may also create one first, save it as a prefab (asset) and then reuse it. In this way, the set components can be saved and this helps to save efforts.
   - **Add script as Component to control pick-ups**: In order to let pick-ups rotate themselves (to make them look more interesting), we add a C# script named Rotator. In this script, we use the class Translate to update the rotation of pick-ups.

f) Collecting and counting
   - **Create GUIText to show text information**: We create a GUIText GameObject and locate it at the upper left corner of the screen.
   - **Update the PlayerController script**: Here we need to update PlayerController script to make the ball capable of being aware of hitting any pick-up. Once it hits a pick-up, we remove the pick-up and update the text in the GUIText GameObject.

g) Publishing the game
   - **Publish the game as it can be player in the Unity Web Player**: File->Build Settings. We set the platform to Web Player and then build the game.