Notes - Unity - Getting Started

- Dr Nick Hayward

A basic intro to getting started with Unity.

Contents

Intro

Unity provides many resources for use and development with their platform. Further details are avilable at the following URL,

- https://unity.com/

The latest version may be downloaded from the following URL,

- https://unity3d.com/get-unity/

Initial learning and tutorial material is also available from Unity,

- https://unity3d.com/learn/tutorials

Getting started

After downloading Unity, we may intsall the plaform with various optional components. For example, we might choose to add build support for the following

- mobile - e.g. Android & iOS
- desktop - e.g. Linux, OS X, & Windows
- TV - e.g Apple's TV OS
- Web - WebGL
- ...

Once installed, we need to create a Unity account, which will be required for development with the platform.

After login, we may create a new project to start initial development. We may define a project name, a local location for storage, either 3D or 2D context for the project, and other various options.

Initial IDE window

As we start a new project, we get a default window with various panes for the project, inspector, and scene.

We can modify this default layout by simply selecting a preferred layout from the ***Layout*** menu in the upper right corner of the IDE window.

A common initial layout is the ***2 by 3*** option, which provides the following panes in the IDE,

- top left - Scene view
    - visually interact, modify, and update the game scene...
    - scene will update relative to game's state
    - elements may be modified, added &c. dynamically as the game progresses - ***n.b.*** changes made whilst playing the game will not be persisted...the game will revert to the default state
- bottom left - Game view
    - player's perspective of the current game state
    - screen aspect ration may be specified or specific type of device, e.g. phone, tablet &c.
    - custom aspect ratios may also be defined
- left vertical - Hierarchy window
    - list of current ***GameObjects*** - empty customisable containers for adding components
    - components may project geometry - e.g. different perspectives, emit light, acts as a camera...
    - ***GameObjects*** may also be hierarchical - e.g. organise multiple GameObjects in a scene
    - ***n.b.*** a given scene may represent a single game level &c.
    - default ***GameObjects*** include a Main Camera and Directional Light
- centre vertical - Project window
    - organises all current game assets - e.g. drag assets from local PC to Project window
    - Unity will automatically update for dragged assets...
- right vertical - Inspector window
    - configure any of the current ***GameObjects***
    - e.g. select an object in the Hierarchy, and properties will be show in Inspector

IDE toolbar

We may also use the IDE's toolbar to develop and manage the game.

e.g. we might add a 3D object from the ***GameObject*** menu, which will be added to the current scene

To pan and drag the current scene, we may use the ***Hand tool*** from the secondary toolbar above the Scene pane. We may also rotate the camera around the current position by using ***right-click*** and ***drag***. The hand icon in the scene will now be updated to an eye.

n.b. on OS X, we may use the ***alt*** key and drag to perform the camera rotate...

Another option is to zoom in and out of the scene using the ***ctlr*** and ***alt*** keys with the mouse. Again, the cursor in the scene will be updated to show a magnifying glass.

The ***Translate tool*** allows us to select and position an object, as required, in the current scene. Guide arrows will also be shown to denote the 3D axes of the current object. These are commonly referred to as a ***gizmo*** relative to Unity development.

n.b. Unity's coordinate system is ***left-handed***. Further information may be found in the following article,

- http://what-when-how.com/xna-game-studio-4-0-programmingdeveloping-for-windows-phone-7-and-xbox-360/3d-math-basics-xna-game-studio-4-0-programming-part-1/

The *Rotate tool* allows us to rotate a game object around the X, Y, and Z axis for 3D development, and the 2D plane as well.

We may also scale a game object using the *Scale tool*. This will scale the object either uniformly on all axes, or separately along the X, Y, and Z axes.

n.b. many of these tools and actions also include keyboard shortcuts for quick access to commands...

gizmo display toggles

These toggles provide additional control to position gizmos in the current scene pane.

These toggles provide support for the following

- Center/Pivot
- Local/Global

In *Center* mode, for selected objects, e.g. two spheres or cubes, a gizmo will be placed equally in the centre of each object. If we then rotate these objects, the centre will be the gizmo in each object. Therefore, uniform rotation will now be possible for these selected objects.

In *Pivot* mode, likewise each selected object will rotate around the gizmo for the equal pivot point.

In *Global* mode, we're able to manipulate the selected object relative to the surrounding world. For example,

- x-axis - goes left to right
- y-axis - goes up and down
- z-axis - goes forward and backward

By contrast, if we switch to *Local* mode, we may manipulate the selected object relative to its own coordinate system. Likewise, the axes will now be modified relative to the game object.

Editor settings

The IDE also provides numerous additional options and settings, which will vary in usage and utility. This will often depend upon the game being developed, and the scale of teams, collaboration &c.

collab

A collaborative option for multi-person development teams.

This option provides a quick and easy way to save, share, and sync a Unity based project amongst disparate team members.

services

Additional services may also be added to a current project, such as

- ads

- analytics
- cloud build
- collaborate
- reports
- in-app purchases
- multiplayer
- ...

n.b. it's necessary to define a project ID prior to activating a service...

layers

Layers may be defined for various specific tasks and restrictions, similar to their use in Adobe's Photoshop app.

For example, a particular layer might be selected to prevent collisions, or perhaps to avoid rendering of specific game objects.