Introduction to unity

Nick Phillips - nicktgphillips@gmail.com

Aim to understand:

- What Unity is
- Unity's interface
- How to add objects into a scene
- How to add scripts (game logic)

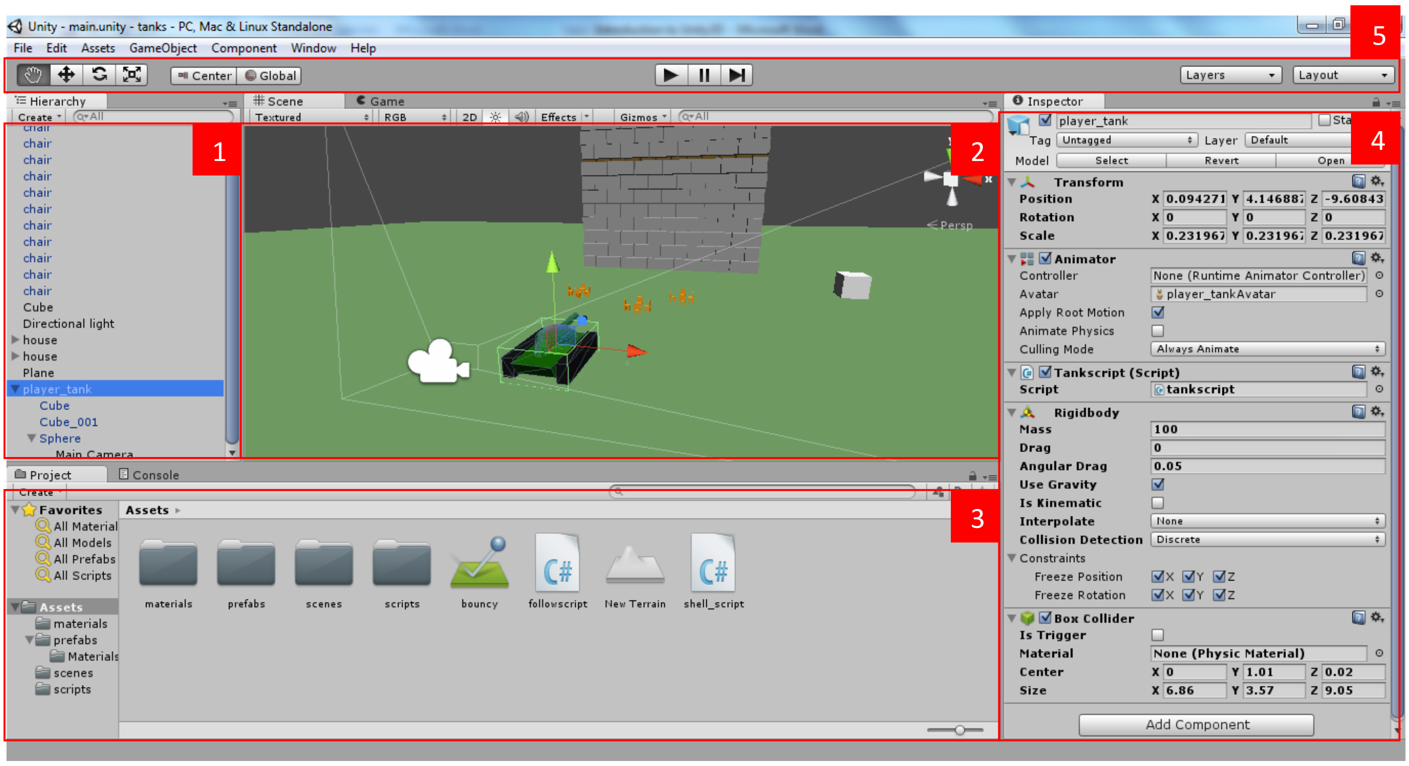
Unity3D (commonly known as Unity) is a very versatile Game Creation engine and environment, which allows programs to be deployed to multiple platforms. Its interface is designed to be user-friendly, and many companies are now using Unity to make their games, including Sony, Blizzard, and Bristol's own Opposable Games. It is now the default software development kit for the Nintendo Wii U. Unity's focus is on creating interactive user experiences, so while video games immediately come to mind, Unity is also being used for more experimental software such as the Oculus Rift VR headset.

Unity Terminology

Since Unity can become quite complex, there is a bit of jargon to explain first

Project - Everything related to your game is kept in a single project space.
Scene - Think of scenes like 'levels' within your game, e.g. MainMenu, Level1, Level2 etc. Your game can have multiple scenes within the same project. You can only work on one scene at a time.
Object - An entity which exists within a scene. e.g. the main camera, light sources, models etc.
View - A part of the unity interface. e.g. Hierarchy view, Scene view, Inspector view etc.

Unity Interface



player_tank
Tag: Untagged | Layer: Default
Model: Select | Revert | Open

Transform
Position X 0.094271 Y 4.146887 Z -9.60843
Rotation X 0 Y 0 Z 0
Scale X 0.231967 Y 0.231967 Z 0.231967

Animator
Controller: None (Runtime Animator Controller)
Avatar: player_tankAvatar
Apply Root Motion: ✓
Animate Physics: ☐
Culling Mode: Always Animate

Tankscript (Script)
Script: tankscript

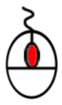
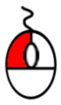
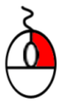
Rigidbody
Mass: 100
Drag: 0
Angular Drag: 0.05
Use Gravity: ✓
Is Kinematic: ☐
Interpolate: None
Collision Detection: Discrete
Constraints
Freeze Position: ✓X ✓Y ✓Z
Freeze Rotation: ✓X ✓Y ✓Z

Box Collider
Is Trigger: ☐
Material: None (Physic Material)
Center X 0 Y 1.01 Z 0.02
Size X 6.86 Y 3.57 Z 9.05

1. **Hierarchy View** - This is where all the objects within your scene are shown. Selected objects will be highlighted
2. **Scene View** - This is the viewport into editing your game. Here you can move around and select objects in your scene
3. **Project View** - This is the folder hierarchy for all of your project's resources. You can also view this same hierarchy using your computer's own file explorer
4. **Inspector View** - This gives a list of all the components attached to the object you have selected. You may edit certain fields shown for each component
5. **Toolbar View** - This holds a variety of controls. On the far left there are transformation tools for manipulating selected objects. To the right of these there are buttons for controlling how objects are manipulated (where their centre is, whether to use global or local axes). In the centre of the toolbar view there are the Play/Pause/Step buttons used for testing out your game. The far right buttons are for configuring the display of layers (sets of objects) and the Unity interface itself

Camera Controls

All of these controls relate to clicking within the **Scene View**.

	- Pan the view
ALT + 	- Rotate the view
ALT + 	- Zoom the view
F	- Focus the view on the selected object. Useful if you have huge scenes

Adding Objects into your Scene - *GameObject -> Create Other*

Unity has a number of different objects you can add into your scene, including particle effects for things like explosions, cameras for multiple viewports, lights to add realism and visual effects, terrain for creating landforms, and more. The models Unity has are very limited, and are basic 3D shapes such as a Cube, Sphere and Cylinder. Unity is meant for game creation, not modelling, so more detailed models are more commonly imported from programs like Blender and Maya.

Clicking on *Create Other* will display all the basic objects Unity can generate itself. When an Object type is clicked, it will appear in the scene.

Hotkeys

Condition	Key/Combination	Action
Object is selected	W	Enables the move tool
	E	Enables the rotate tool
	R	Enables the scale tool
	CTRL D	Duplicates the selected object
	DEL	Deletes the selected object
	CTRL Z	Undo
	CTRL Y	Redo
	CTRL P	Play
	CTRL SHIFT P	Pause

Scripts - *Select Object, Inspector View -> Add Component -> New Script (Choose C# and name)*

Scripts are the pieces of code which define your game's functionality. They are used for adding control and gameplay behaviour. Scripts are treated as components, and as such, they must be added to an object in order for them to be executed.

By default, scripts are edited using MonoDevelop, a program which is installed alongside Unity. When you open a newly-created script, you will see that there are sections of code written for you, acting as a template for you to place your own code.

```
using UnityEngine;
using System.Collections;

public class MyScript : MonoBehaviour {

	// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}
}
```

`Start()` is executed when the object is first created. If the object already exists in your scene at the beginning, the code within the curly brackets is run when you first open/enter the scene. This function is good for initialising variables, especially references to other GameObjects.

`Update()` is executed every single frame. For a game with a frame rate of 30 frames per second, this means that this script will be run 30 times per second. Because it is looping, this function is useful for detecting user inputs (keyboard presses) and checking physical properties of objects (itself included).

Moving an object one unit in the 'z' axis when 'W' key is pressed (inside `Update()`):

```
if(Input.GetKey(KeyCode.W)){
      transform.Translate(0,0,1);
}
```

Adding Physics!

To make an object react (kinda) realistically to conventional physics, it must have both a Collider component and a RigidBody component. By default, Unity's built-in physical models have Colliders, so you don't have to worry about adding one. You do however need to add a RigidBody component.

The Collider component makes objects physically solid. Without a RigidBody component, the object will just stay fixed in position if another object collides with it.

The RigidBody component enables the object to move under physical forces. you may also edit its mass, turn gravity on and off, and edit other options through the inspector view.

Here is some code for jumping in the y-axis with a force of 100 units (inside `Update` function, remember to add a RigidBody component to your object!):

```
if(Input.GetKey(KeyCode.Space)){
      rigidbody.AddForce(0,100,0);
}
```

Learning at Home

If you thought this workshop went too fast, or you don't think we've gotten to the interesting bits about Unity, don't worry! You can carry on learning at home.

Here are a few resources which I found useful (roughly in the order of what I use when I am stuck):

- Unity's API (list of functions and objects with descriptions)
    - http://docs.unity3d.com/ScriptReference/
- Google
    - Just type "Unity" followed by what you're looking for. If you have a question, it is very likely that it has been discussed and answered on a forum. Google is the best resource for finding these forum posts
- YouTube
    - There are thousands of Unity tutorials on here, both official and unofficial. Most will offer you good advice about game design and guide you through everything from the basics to developing an RPG game

As a final tip, **always remember to check your curly brackets!!**

I hope you've enjoyed the workshop!

- Nick Phillips (nicktgphillips@gmail.com)