Unity Pro

CMPS179 Game Design Practicum Lecture 7

Content mostly taken from Unity website and tutorials 😊

Some slides taken wholesale from Microsoft Kinect SDK Documentation
Unity from 10,000 feet

• An IDE designed to accelerate game design
• Composes and describes relationships between assets, scripts and “scenes”
• Allows rapid development cycles, live previews and tweaking while game runs
• Extensible (see GAK plugin)
Why learn Unity?

• Can publish to a wide variety of platforms, including iPhone and Xbox 360
• Useful for prototyping a design quickly, as many things (particles, collisions, and camera controls) have already been done for you
• Games created with unity include: Battlestar Galactica Online, Runespell, etc
What Unity Doesn’t Do

- Computer Vision
- Really Cool Things Unthought Of ®
- Experimental gameplay (Not traditional first/third/etc. Basically Kinect)
- Depth/image information processing
Unity Video Demo

• What can unity *really* do?
THE UNITY INTERFACE
Interface Layout
Scene

- Navigation
- Edit GameObjects
- Scale/Rotate/Transform
Navigation in Scene View

- Hold the right mouse button to enter **Flythrough** mode. This turns your mouse and **WASD** keys (plus **Q** and **E** for up and down) into quick first-person view navigation.
- Select any GameObject and press the **F** key. This will center the Scene View and pivot point on the selection.
- Use the arrow keys to move around on the X/Z plane.
- Hold **Alt** and click-drag to orbit the camera around the current pivot point.
- Hold **Alt** and middle click-drag to drag the Scene View camera around.
- Hold **Alt** and right click-drag to zoom the Scene View. This is the same as scrolling with your mouse wheel.
Positioning Game Objects

- Positioning GameObjects
- See [Positioning GameObjects](http://eis.ucsc.edu) for full details on positioning GameObjects in the scene. Here's a brief overview of the essentials:

![Images of Translate, Rotate, and Scale functions]
Inspector

• Used for adjusting components
• Relationship between GameObjects and Components shown here
• Can enable/disable components and adjust parameters
Hierarchy

- Some of these are direct instances of asset files like 3D models, and others are instances of Prefabs -- custom objects that will make up much of your game.
- You can select and Parent objects in the Hierarchy.
- As objects are added and removed from the scene, they will appear and disappear from the Hierarchy as well.
Transform component

Don’t scale! Do scaling in Blender/Etc.

**Parenting**
When a GameObject is a **Parent** of another GameObject, the **Child** GameObject will move, rotate, and scale exactly as its Parent does. Just like your arms are attached to your body, when you turn your body, your arms move because they're attached. Any object can have multiple children, but only one parent.
A note on Parenting and Transforms

• When parenting Transforms, set the parent's location to <0,0,0> before adding the child. This will save you many headaches later.

• **Particle Systems** are not affected by the Transform's Scale. In order to scale a Particle System, you need to modify the properties in the System's Particle Emitter, Animator and Renderer.
Game View
Live preview of game
Game View Notes

• Any changes won’t be baked in when game is running, need to reapply them
• Will reflect current position/state of objects in scene view as well
CONCEPTS AND ELEMENTS
GameObjects

• These are everything in Unity – are represented in various ways in interface and in the code
• Can navigate between them using scripts
Cameras

• See Live Demo
Particle Systems/Physics

• Collidable
SCRIPTING
Scripting

- **Scripting Overview**
- Scripting inside Unity consists of attaching custom script objects called behaviours to game objects.
- Different functions inside the script objects are called on certain events.
- **Update:**
  - This function is called before rendering a frame.
  - This is where most game behaviour code goes, except physics code.
- **FixedUpdate:**
  - This function is called once every physics time step.
  - This is the place to do physics-based game behaviour.
- **Code outside any function:**
  - Code outside functions is run when the object is loaded.
  - This can be used to initialise the state of the script.
Examples of editing properties

using UnityEngine;
using System.Collections;

public class example : MonoBehaviour {
    void Update() {
        transform.Translate(0, 0, 2);
    }
}
Timing

```csharp
public class example : MonoBehaviour {
    void Update() {
        transform.Rotate(0, 5 * Time.deltaTime, 0);
    }
}
```
public class example : MonoBehaviour {
    public Transform explosion;
    void OnCollisionEnter() {
        Destroy(gameObject);
        Transform theClonedExplosion;
        theClonedExplosion = Instantiate(explosion, transform.position, transform.rotation) as Transform;
    }
}
Yield

public class example : MonoBehaviour {
    IEnumerator Do() {
        print("Do now");
        yield return new WaitForSeconds(2);
        print("Do 2 seconds later");
    }
    void Example() {
        Do();
        print("This is printed immediately");
    }
}
Working with Kinect Data

• Same data – skeleton joints as
LIVE DEMO
Unity Pro

John Murray