COMPSCI 715
Advanced Computer Graphics

More Unity

Schedule 1st Half

Week	Activities	Assignments
1	Tue: course outline, Wed: project topics, Thu: feedforward learning (guest lecture)	Register teams
2	Tue + Thu: exergaming (guest lectures), Wed: academic writing overview, abstracts	Abstract (1.5%)
3	Tue + Thu: Unity, Wed: writing introductions	Introduction (2.5%)
4	Tue + Thu: Unity, Wed: writing about related work	Related work (2.5%)
5	Team meetings, Wed: design & implement.	1st prototype (1.5%)
6	Team meetings, Wed + Thu: demos (2.5%)	2nd prototype (1.5 %)

Mid-semester break. So far 12% of individual assignments.

Today's Mission

Using Unity...

1. How to to create objects dynamically?
2. How do simple animations work?
3. How to support the Oculus and the Kinect?

Recap: Script Example
Use Input to Control Physics

- GetComponent<ComponentType> () to get a certain component of this GameObject
  - Slow, so better not in (Fixed)Update
- Add force in FixedUpdate (!)
  - Force causes acceleration
  - Experiment with values to find right one
- Limit speed by checking & truncating length of velocity
  - But this will also limit effect of gravity, explosions etc.

```
public class PlayerControl : MonoBehaviour {
   float accel = 5f;       // acceleration
   float maxSpd = 2f;  // maximum speed (m/s)
   Rigidbody rigidBody;
   void Start() {
      rigidBody = GetComponent<Rigidbody> ();
   }
   void FixedUpdate () {
     rigidBody.AddForce (
        Input.GetAxis("Horizontal") * accel,
        0,
        Input.GetAxis("Vertical") * accel);
     if(rigidBody.velocity.magnitude > maxSpd)
       rigidBody.velocity =
        rigidbody.velocity.normalized *
maxSpd;
   }
}
```

Script Example: Falling Rocks

- Invoke for timed method calls
- Instantiate and Destroy for cloning/destroying objects
- Use objects as prefabs to instantiate other objects (drag to/from Assets/Prefabs)

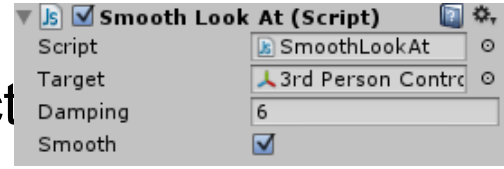
```
public class Destroyer2 : MonoBehaviour {
   public GameObject explosion;
   void Start () {
     Invoke ("Boom", 6);
     Destroy(gameObject, 6.1f);
   }
   void Boom () {
       Instantiate (explosion,
           gameObject.transform.position,
           Quaternion.identity);
} }
```

```
public class Instantiator : MonoBehaviour {
   public GameObject target;  // “prefab”
   void Start() {
       InvokeRepeating("SpawnObject", 1, 2);
   }   // invoke after 1 sec every 2 secs
   void SpawnObject() {
       float x = Random.Range(-2.0f, 2.0f);
       float z = Random.Range(-2.0f, 2.0f);
       Instantiate(target, new Vector3(x, 10, z),
           Quaternion.identity);  // global coords
   }
}
```

```
public class Destroyer : MonoBehaviour {
   void Start() {
       Destroy(gameObject, 6);  // 6 sec delay
   }
}
```

Reusing Existing Assets

- Import assets from Asset Store: Window -> Asset Store
- Or import from standard packages: Assets -> Import Package
  Find them in: Project Assets \ Standard Assets
  - Character Controllers \ First Person Controller
    Just as you know it from FPSs...
  - Character Controllers \ 3rd Person Controller
    Combines animated character with an Animation, ThirdPersonController and ThirdPersonCamera
  - Camera Scripts \ SmoothLookAt
    Camera smoothly turns toward target object
  - Camera Scripts \ SmoothFollow
    Camera smoothly follows the set target object



Animation

- Use Animation component
  - Add Animations to use
  - Use Play() method on Animation component
- Advanced: for smooth transitions between animations, use Animator and BlendTree

```
public class AnimationPlay : MonoBehaviour {
   void Update() {
        if (Input.GetAxis("Horizontal") == 0
          && Input.GetAxis("Vertical") == 0)
            animation.Play("idle");
        else
            animation.Play("run");
}  }
```

- Or get and change transform of object directly
- Forward Kinematics: change rotation of limb to achieve a pose
- Advanced: With Inverse Kinematics (IK) change position of limb and rotations are calculated automatically

```
public class RotateTarget : MonoBehaviour {
   public Transform target;
   public float speed;
   void Update () {
     if (Input.GetButton ("Fire1")) // left Ctrl
       target.eulerAngles += new Vector3(
           0, Time.deltaTime * speed, 0);
}  }
```

Using the Oculus Rift

1. Download Unity Integration asset package (need to register): https://developer.oculusvr.com/
2. Import it: Assets -> Import Package -> Custom Package… OculusUnityIntegration.unitypackage
3. Drag Oculus VR prefab from Project tab OVR \ Prefabs into Scene or Hierarchy to use it:
   - OVRCameraController: normal camera for Oculus
   - OVRPlayerController: first-person player object
4. Make sure non-Oculus compatible objects are removed
   - Normal camera should be removed
   - 2D objects (GUI etc.) should be replaced by 3D objects

Using the Kinect

- Use SkeletonWrapper component: http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK
- To decouple Kinect (runs only at 30 frames/sec) from Unity rendering, poll on new thread
- setupFinished indicates whether Kinect calibration has finished
- pollSkeleton() get new skeleton positions
- bonePos array contains current bone positions for all players
  - 1st array dimension for player number
  - 2nd dimension: Use enum NuiSkeletonPositionIndex to get index for particular bone

```
public class KinectController: MonoBehaviour{
   public SkeletonWrapper sw;
   public int player = 0;
   public Transform head;
   void Start() {
      new Thread(Run).Start();
   }
   void Run() {
     Debug.Log("Kinect Thread Started");
     Thread.Sleep(50);
     if (sw.setupFinished && sw.pollSkeleton())
{
       head.x = sw.bonePos[player,
           NuiSkeletonPositionIndex.Head].x;
        head.y = sw.bonePos[player,
           NuiSkeletonPositionIndex.Head].y;
        head.z = sw.bonePos[player,
           NuiSkeletonPositionIndex.Head].z;
 }}}}
```