**32004 Game Development**

Assessment

1. Programming Exercises: 30% weekly

2. Quiz: 15% week10 (open book)

3. Classic Game Recreation(娱乐，消遣)-Getting Started: 20% week7

4. Classic Game Recreation-Enhancing and Innovation: 35% week12

**Week 2**

1. Camera:

A window into the game’s scene that represents the player’s view

– What the camera sees is what the player sees

– Where the camera goes is where the player goes

A camera is just a game object with a Camera component

– You can have multiple cameras in one scene

– Either with only one active at a time OR all active and rendering(着色, 这里翻译成渲染) to different parts of the player’s screen

– Good for splitscreen(分画面) multiplayer, rendering minimaps, and overlaying(覆盖) weapons(武器) in first person view

1.1 Camera Properties

Clear Flags

Each Camera stores color and depth information when it renders its view. The portions of the screen that are not drawn in are empty, and will display the skybox by default. When you are using multiple Cameras, each one stores its own color and depth information in buffers, accumulating more data as each Camera renders. As any particular Camera in your scene renders its view, you can set the Clear Flags to clear different collections of the buffer information. This is done by choosing one of the four options:

[每个相机在渲染时会存储颜色和深度信息。屏幕的未绘制部分是空的，默认情况下会显示天空盒。当你使用相机时，每一个都将自己的颜色和深度信息存储在缓冲区中，还将积累大量的每个相机的渲染数据。当场景中的任何特定相机进行渲染时，你可以设定清除标记以清除缓冲区信息的不同集合。可以通过下面四个四个选项之一来完成

Skybox

This is the default setting. Any empty portions of the screen will display the current Camera's skybox. If the current Camera has no skybox set, it will default to the skybox chosen in the Render Settings (found in Edit->Render Settings). It will then fall back to the Background Color. Alternatively a Skybox component can be added to the camera

Solid Color 纯色

Any empty portions of the screen will display the current Camera's Background Color.

Clipping planes [　建材平面：从相机到开始渲染和停止渲染之间的距离。]

– Near plane – anything closer to the camera than this distance will not be rendered

– Far plane – anything further than this plane will not be rendered

1.2 Camera projection [相机投影]

Perspective[透视]:近大远小，有距离之分

Orthographic[正交相机]: 没有近大远小距离之分，如两个一样大的物体放在相机相同位置，或者有远近的放置，是看不出来大小缩放的，也看不出谁前谁后。

1. Lighting [详见：<https://blog.csdn.net/qq_30687901/article/details/53792155]>
2. 3D Models

Unity has 6 primitives: cube, sphere,plane, capsule, cylinder, quad

1. Materials

4.1Color, texture, light bounce

Shader [着色器]: code that dictates how a material is applied to a mesh

Albedo[反照率]: 256是峰值

Metallic and Smoothness: how light distributes over the surface of the material [光是否均匀的分布在材料]

4.2 normal map, bump map, height map

4.3 particle effects [颗粒特效]

**Week 3**

1. Animation : An Animator Component uses an Animator Controller

An Animator Controller organizes multiple Animations

Animations are always just a series of images, shown in rapid succession, to give thee illusion of motion.

**Week 6**

1. Frame & Fixed Frame

注释：

FixedUpdate () 和 Update ()

同：当MonoBehaviour启用时，其在每一帧被调用。都是用来更新的

异：Update()每一帧的时间不固定，即第一帧与第二帧的时间t1和第三帧与第四帧的时间t2不一定相同。FixedUpdate()每帧与每帧之间相差的时间是固定的.

Update受当前渲染的物体影响，这与当前场景中正在被渲染的物体有关（比如人物的面数，个数等），有时快有时慢，帧率会变化，Update被调用的时间间隔就会发生变化。但是FixedUpdate则不受帧率的变化影响，它是以固定的时间间隔来被调用。

所以一些物理属性的更新操作应该放在FxiedUpdate中操作，比如Force，Collider，Rigidbody等。外设的操作也是，比如说键盘或者鼠标的输入输出Input，因为这样GameObject的物理表现的更平滑，更接近现实。

FixedUpdate的时间间隔可以在项目设置中更改，Edit->Project Setting->time 找到Fixed timestep。就可以修改了

1. Time.类 (详解：https://blog.csdn.net/u010377179/article/details/52618291 )

Time.time : 从游戏开始到当前的时间

Time.timeSinceLevelLoad : 从这个场景载入到当前的时间

Time.unscaledTime :（只读）不考虑timescale时候与time相同，若timescale被设置，则无效。

1. Delta Time：每一次／自上一次Update（）调用以来经过的时间量；是可以改变的／不同设备之间也会有差异

Update()更新之间的时间叫做delta time(); FixedUpDate()更新之间的时间叫做fixed delta time();

4. Time.timeScale: Controls the speed (the “scale”) at which time passes (取值在0-100)

– 1.0f = realtime； – 0.5f = half of realtime； – 0.0f = time is not updated, “paused”

1. Invoke:

E.g. player presses Fire1, detonate bomb 1.5 seconds later:

if (Input.GetKeyDown(KeyCode.space) Invoke(“DetonateBomb”, 1.5f);

Invoke() 方法是 Unity3D 的一种委托机制

如： Invoke("SendMsg", 5); 它的意思是：5 秒之后调用 SendMsg() 方法；

使用 Invoke() 方法需要注意 3点：

1 ：它应该在 脚本的生命周期里的（Start、Update、OnGUI、FixedUpdate、LateUpdate）中被调用；

2：Invoke(); 不能接受含有 参数的方法；

3：在 Time.ScaleTime = 0; 时， Invoke() 无效，因为它不会被调用到

Invoke() 也支持重复调用：

InvokeRepeating("SendMsg", 2 , 3);

这个方法的意思是指：2 秒后调用 SendMsg() 方法，并且之后每隔 3 秒调用一次 SendMsg () 方法

1. Coroutines 协同程序

注释 : – Unity’s equivalent of multi-threading (but not really muti-threading)

– Allows decoupling(分离的) from main game loop

void Update() {

StartCoroutine(DoStuff());

OtherMethod();}

IEnumerator DoStuff() {

Debug.Log(“Doing all the stuff”);

yield return null; // Wait until the next frame before continuing

Yield return new WaitForSeconds(1.0f); // wait for 1s

Debug.Log(“Doing more stuff”);}