# U3D

## UNITY

### 配置

============安装

Browse

C:\Program Files\Unity\Editor

点击Patch

=========打开

edit->preference->General

alaways show project wizard打勾

打开时，选择工程的文件夹就是打开工程，或者可以打开Assets文件中保存的后缀为unity的场景文件。

==========andriod发布

Build settings

scene in build： add current

player settings （andriod）设置：com.test.test

=========编辑器设置

Layout设为tall

Edit-》preference-》external tools-》external script editor，可以选择vs的编辑器。

=========导入包

如果包中包含.unity后缀的场景文件，可以打开一个场景。场景是使用各种元素搭建起来的工程。

========文件路径不能包含中文

### 视图

Z轴（蓝色）代表相机视角的方向。Y代表空间高度（上下，可以用0为基准）。X代表左右。

点击坐标轴，可以改变场景视图，用二维的视角进行调节。

按住alt和鼠标左键，可以任意移动场景视图角度。

Scene上面有四个按钮（对应快捷键QWER）。

第一个按钮手，代表移动场景视图。

第二个十字形，代表在鼠标场景中是移动模型的位置（拖动箭头可以单方向移动，选中的箭头会变成黄色）。

第三个代表旋转，选择了模型后，点击选中的轴线，沿着线移动（线平行方向移动，不是垂直方向）。

第四个代表缩放操作。

### Scene

File->New Scene(Ctrl+N)可以创建不同场景。每个场景都是独立的，可以使用相同的资源，如模型，代码等。

Build Setting中，可以将当前场景加入。在其中加入多个场景，就可以在代码中切换场景。放在最上面的场景是游戏刚开始运行时出现的场景。

Application.LoadLevel("s1");//s1代表某个场景的名称

### Assets(project)

该菜单项目对应Assets视图中鼠标右键菜单。

选择Create，可以加入各种资源。

#### Prefab

Prefab相当与一个将各种模型，代码等组织在一起的游戏物体（GameObject）。

将hierarchy中的模型直接拖入Asset文件夹中，可以直接生成prefab。

也可以新建一个prefab，将模型或者代码文件拖拽到prefab中。

使用时，可以直接拖入场景，也可以拖入Hierarchy视图中。

#### Material

可以为场景中的模型添加颜色与贴图，设置该资源的属性，拖入场景中的模型，即可改变模型的样子。

### GameObject(hierarchy)

该菜单项目可以对应hierarchy视图中的create。

可以直接加入物体到了场景中（不加入到Assets中）

所有游戏物体都有Transform属性，其他属性都可以删除或者添加。即所有游戏物体都是在transform属性基础上叠加其他属性。

#### Directional light

可以加入光源，否则场景会很暗。

#### Plane

默认出现的位置相当于地面。

直接将图片拖拽到该plane即可改变。

可以用来当墙面，或者GUI的背景。

#### GUI Texture

类似与plane，默认出现位置相当于相机的对面。

可以向Inspecror中的Texture栏目拖入图片。

可以为这个GUI Texture增加脚本，按实际情况设置位置

void Start () {

guiTexture.pixelInset = new Rect(0, 0, Screen.width, Screen.height);

}

### Component(Inspector)

该菜单项目可以对应inspector视图中的add Component。

Inspector视图中包含了模型的各种属性。

#### 天空

选中相机， rendering-》Sky Box，然后可以在相机对应的inspector视图中看到sky box，点击custom skybox那行的按钮，即可选择天空(需要先导入天空的包)

#### 刚体

Obj是GameObject，而且包含刚体属性

##### 重力

physicals-》rigibody，物体会自动往地面落下。

Constrains-》freeze rotation，勾选x，y，z。则落下时不会改变旋转位置。也可以加入代码

if (rigidbody) rigidbody.freezeRotation = true;

rigidbody.useGravity = false;

Physics.gravity.y = -2;

##### 加速度

//可以让刚体朝着前方运动，初速度是200（速度的矢量方向合成）

Vector3 v = new Vector3(0,0,200);

Obj.rigidbody.velocity= transform.TransformDirection(v);

##### 力矩（自旋转）

Obj.rigidbody.AddTorque(Random.onUnitSphere \* 0.1f,ForceMode.Impulse);

#### 碰撞

当两个模型接触后，如果都有碰撞体（且物体是刚体)，就会有碰撞效果。

component-》physicals加入Box Collider等。

做碰撞效果时，刚体的属性不要约束位置的x，y，z，否则会在碰撞后一直移动。

#### 隐藏

去掉Mesh Renderer属性，可以让模型看不见

#### 声音

音频监听器（Audio listener）通常附加在使用的相机上（在加入Audio Source会自动加入到相机）。声音在3D空间内，所以如果包含声音的模型远离相机，声音就会减小。

勾选play on Awake可以在物体出现时自动播放，其他属性都可以在这个Audio Source中设置。

void Awake() {

audio.volume = 0.2F;

audio.Play();

}

播放assets中的声音

模型必须有Audio Source

public AudioClip sound=new AudioClip(); //可以在inspector视图中选择声音

audio.PlayOneShot (sound, 1);//1代表音量最大

#### Animation

模型中包含Animation组件

animation.Play("\*\*\*", PlayMode.StopAll);

如果没有在播放任何动画，就播放

if (!animation.isPlaying)

animation.Play("\*\*\*");

### 代码编辑

project中assets区域，右键，Create，C#script（还可以导入package，将其中的资源直接拖入左边的场景中，或拖拽到hierarcy视图中的物体上）

能拖入物体的代码必须继承自MonoBehaviour，Update,FixedUpdate函数是按照帧频自动调用。

### 位置移动

移动当前模型的位置

if(Input.GetKey(KeyCode.W))

{//写成Vector3.forward\*0.1f，则可以减少移动的大小，不写\*0.1f会自动赋值一个移动大小

transform.Translate(Vector3.forward);

}

if(Input.GetKey(KeyCode.S))

{

transform.Translate(Vector3.back);

}

if (Input.GetKey (KeyCode.A))

{

transform.Translate(Vector3.left);

}

if (Input.GetKey (KeyCode.D))

{

transform.Translate( Vector3.right);

}

游戏开始时，将当前模型在1s内移动到目标模型的位置

transform.position = Vector3.Lerp(transform.position, target.transform.position, Time.time);//Time.time指游戏开始到现在所用的时间

### 输入设备

Edit->project Settings->input,在视图中可以看到默认的输入（Axes）

右键输入项目，可以选择复制或者删除输入项目

float WorS = 0.0f;//-1...1

WorS=Input.GetAxis("Vertical");//对应Axes中的名称

if (WorS > 0)

{

transform.Translate(Vector3.forward\*0.1f);

}

else if(WorS<0)

{

transform.Translate(Vector3.back\*0.1f);

}

----------------------------------------THRUSTMASTER-------------------------------------------

(必要时删除多余输入设置，只使用手柄的输入设置，避免按键定义干扰)

=====左右

Type： Joystick Axis

Axis：X axis

float x = Input.GetAxis("Horizontal");

=====上下

Type： Joystick Axis

Axis: Y axies

float y = Input.GetAxis("Vertical");

=====滑动轴

Type： Joystick Axis

Dead：0.1

Sensitivity:1

Axis: 4th axis(Joysticks)

float x = Input.GetAxis("\*\*");

=========按钮(使用GetButtonUp或GetButton)

-----前按钮

if (Input.GetButton("\*\*"))

Positive Button：joystick button 0

-----中按钮

Positive Button：joystick button 1

-----左按钮

Positive Button：joystick button 2

-----右按钮

Positive Button：joystick button 3

### Camera

场景中必须有一个相机对象，且camera组件被勾选。

代码中Camera.main指当前使用的相机

相机切换

public Camera camera0，camera1;

camera0.enabled = true;

camera1.enabled = false;

### 视角转动

可以移动视角（相机跟随物体）, 可以直接放到update中，相当于鼠标移动。

transform.Rotate(0, Input.GetAxis("Mouse X"), 0);

transform.Rotate(-Input.GetAxis("Mouse Y"), 0, 0);

转动位置设置

transform.rotation = new Quaternion(0, 0, 0, transform.rotation.w);

限制上下的视角转动（transform.rotation.x总是小数）

if (transform.rotation.x < 0.2&&transform.rotation.x>-0.2)

{

transform.Rotate(-Input.GetAxis("Mouse Y"), 0, 0);

}

if(transform.rotation.x > 0.2)

{

transform.Rotate(-1, 0, 0);

}

else if (transform.rotation.x < -0.2)

{

transform.Rotate(1, 0, 0);

}

### 坐标转换

==========世界坐标系转屏幕坐标系

Vector3 Pos = Camera.main.WorldToScreenPoint(transform.position);

==========屏幕坐标系转世界坐标系

Vector3 v = Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, Camera.main.nearClipPlane+40));

可以将屏幕坐标转换为世界坐标，Camera.main.nearClipPlane+40可以让z的位置在相机面前的40m远处, Camera.main.nearClipPlane代表相机所在的Z平面。

------物体移动到鼠标点击的位置（假设代码放在物体中）

float near = transform.position.z-Camera.main.transform.position.z;

Vector3 v = Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, Camera.main.nearClipPlane+near));

transform.position = new Vector3(v.x, v.y,transform.position.z);

### 模型控制

#### 模型变量

public GameObject[] obj ;//可以在Inspector视图将物体拖入，Inspector上面关于该数组的size会自动变化。如果是int这种类型，可以在size中输入大小，回车确认，然后进行填写。

在inspector视图中找到该变量，选择物体

在C#脚本中声明了public变量，将脚本加入GameObject，则可以在该模型的Inspector中代码文件放置的位置看到该变量，而且可用在这里设置初始值的大小。

通过使用函数查找

public GameObject treeObj;//类内变量

treeObj=GameObject.Find ("tree");//放在Start函数中

tree是要控制的模型的名字，通过treeObj可以对模型进行控制

#### 组件

A a= GetComponentsInChildren<A>();

可以获取子物体组件，即物体中包含的子物体中如果包含的代码中有类A（或继承自类A），就可以获取子物体。

Obj.GetComponent<AudioListener> ().enabled = false;

可以让模型中的声音侦听器不可用

#### 复制

GameObject clone = (GameObject)Instantiate(Obj, new Vector3(x, y,z), transform.rotation);

将Obj进行复制，位置为（x,y,z）,角度为当前的物件角度（可以使用Quaternion.identity，让角度为默认的直线）。

#### 碰撞销毁

在两个物体碰撞时，将另一个物体（没有加入这个代码的物体）销毁。

将碰撞体的属性Is Trigger打勾。  
void OnTriggerEnter ( Collider other ){Destroy(other.gameObject);}

在检测到碰撞后销毁当前物体

void OnCollisionEnter(Collision collision)

{

if (collision.gameObject.tag == "\*\*\*")//可以为被碰撞的物体设置标签，当前物体碰撞另一个物体时，通过标签判断是不是碰撞到了指定物体

Destroy(gameObject);

}

#### 坐标位置

Obj.transform.position=new Vector3(

transform.position.x,

transform.position.y,

transform.position.z-1f);

//JS写法 var direction = Vector3(0,5,0);

在hierarchicy视图中，将模型B拖入模型A下层，则可以让B随着A位置自动移动，此时B的坐标是相对与A的位置的坐标。

#### 旋转

默认的角度Quaternion.identity

转动时遵守左手螺旋定则。

Obj.transform.Rotate(0, 100\* Time.deltaTime, 0);

//自旋转，绕y轴每秒转100度

obj.transform.RotateAround (obj2.transform.position, Vector3.up, 80 \* Time.deltaTime);

//围绕obj2，每秒转80度

transform.rotation = Quaternion.identity;

//重置旋转角度

### 外部命令

[RequireComponent(typeof(Rigidbody))]

可以在代码中加入模型属性，确保在视图中不能删除该属性

[HideInInspector]

代码中下面一行的public变量不会在视图中出现

[AddComponentMenu("\*\*\*")]

可以在Component菜单中增加菜单项目。

在Hierarchy视图中选择中物体后，点击菜单，就可以将代码文件直接加入到物体。

[SerializeField]

private int x=0；

私有变量不在Inspector视图中显示，但是加了SerializeField就可以在视图中显示了。

### EditorWindow

在项目中放入继承自EditorWindow的代码，写入OnGUI这样的函数，就可以在编辑窗口中增加一个自定义的菜单项目。

## GUI

加入函数，将GUI内容写在该函数内

void OnGUI()

### 层级

GUI.depth = -100;

### 颜色

Color.clear;代表纯透明

Color color= Color.red;

color.a = 0.5f;

//自定义颜色，可以设置颜色的GRBA，a设置0.5代表半透明，0代表纯透明

public Color color = new Color(0.2F, 0.3F, 0.4F，0.5F);//GRBA

var color : Color = Color(0.2, 0.3, 0.4);

可以改变控件的颜色,该类代码一定要放在OnGUI() 函数最上面。

[GUI.color](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/GUI/GUI.color.html) = [Color.yellow](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/Color/Color.yellow.html);// 将影响背景和文本颜色。

GUI.backgroundColor = Color.red;//背景颜色

GUI.contentColor = Color.yellow;//文本颜色

### 字体大小

以设置button为例

方法一：（会影响整个程序的字体）

GUI.skin.button.fontSize=30;

方法二：（只设置单独的实例）

GUIStyle style=new GUIStyle();

style.normal.background = null;

style.normal.textColor=new Color(1,0,0);

style.fontSize = 40;

GUI.Button (new Rect (x, h , width, h), "nn",style)

### 位置

GUI.skin.label.alignment = TextAnchor.UpperRight;

可以让label在右上角

### 按钮

if (GUI.Button (new Rect (x, y, cx, cy), "name")) {

[Debug.Log](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/Debug/Debug.Log.html)("Clicked the button with an image");

}

### 输入文本

private string str=”xxxx”;

str = GUI.TextArea(new Rect(x, y, cx, cy), str, 500);

### 静态文本

[GUI.Label](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/GUI/GUI.Label.html)(new [Rect](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/Rect/Rect.html)(x, y, cx, cy), "Hello World!");

### 图片绘制

public Texture2D cur;//Inspector中选择图片

GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height), cur, ScaleMode.StretchToFill, true, 10.0F);//最后两个参数（进行通道融合，缩放比例10：1），可以不写

### 视频

public MovieTexture movTexture;//视图中拖入视频，需要安装quicktime解码

GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height), movTexture, ScaleMode.StretchToFill);

movTexture.Play();//可以写在其他函数中

## function

### 定时器

Invoke("fun", 2);//2秒后执行fun函数

void fun() { print("xx");}

### 键盘/鼠标

if (Input.GetMouseButton(0))//判断鼠标左键是否按下，参数1时，代表右键

Screen.showCursor = false; //隐藏鼠标

Screen.lockCursor=true;

将鼠标隐藏并锁定在视图中间

鼠标位置

如果获取的位置是以左下角为原点，需要进行转换

public float x = Input.mousePosition.x;

public float y = Screen.height-Input.mousePosition.y;

### 退出程序

Application.Quit();

### 分辨率

int cx=Screen.width;

int cy=Screen.height;

### 限制数值

限制value的值在min和max之间， 如果value小于min，返回min。 如果value大于max，返回max，否则返回value

*Clamp* (*value* : float, *min* : float, *max* : float)

### 两点间距离

float x = Vector3.Distance(a : Vector3, b : Vector3);

### 调试

[Debug.Log](mk:@MSITStore:D:\\fileFFFFFFFFFFFFFFFFFFFFFF\\JYL\\文件\\unity3d中文api.chm::/Debug/Debug.Log.html)("xx");

print(Input.mousePosition);

### 定时器

float timer=3.0f;

update中：

timer -= Time.deltaTime;

### 随机

Random.value //得到0到1之间的随机数

if ( Random.Range(0,100)<30)//在范围内取随机数，概率为30%

Random.rotation

### 鼠标点击移动

会自动寻找鼠标点击的方向中的碰撞体，找到了碰撞体，就会往碰撞体定义一个射线

类内变量

public Vector3 end;

初始化

end = transform.position;

update函数中

if (Input.GetMouseButtonDown(0))

{

//定义一个当前相机到另一个物体之间的射线。

Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

RaycastHit hit;

if (Physics.Raycast(ray, out hit))

{

end = hit.point;

}

}

transform.position = Vector3.Lerp(transform.position, end, Time.deltaTime \* 2.0f);