Unity游戏编程基础教学

Fundamentals of Game

Programming with Unity

Unity常用API

- Unity常用基础类
- Unity其它功能类

Unity常用基础类

- GameObject
- Transform
- Vector3
- Time
- Input
- Mathf
- Random

GameObject

GameObject类常用于处理游戏对象共同拥有的属性，同时GameObject类可以用来寻找特定的游戏对象，以及添加、获取组件等功能。

GameObject公有字段：

```
public string tag                    //该游戏对象的标签名。
public Transform transform           //访问该游戏对象的Transform属性。
```

GameObject静态函数：

```
//创建一个类型为type，且拥有对应网格渲染器和碰撞体组件的游戏对象。
public static GameObject CreatePrimitive(PrimitiveType type);
//其中：
//PrimitiveType枚举变量包括：Sphere、Capsule、Cylinder、Cube、Plane、Quad

//寻找游戏对象（深度优先）
public static GameObject Find(string name);          //根据名字name寻找单个游戏对象
public static GameObject FindWithTag(string tag);    //根据标签tag寻找单个游戏对象

//根据标签寻找所有游戏对象（根据深度优先顺序依次编号）
public static GameObject[] FindGameObjectsWithTag(string tag);
```

GameObject

GameObject公有函数：

//为游戏对象添加组件，参数classname表示组件类型。
public Component AddComponent(string className);
//注意使用as转换类型：gameobject.AddComponent(“rigidbody”) as rigidbody。

//设置游戏对象的激活状态：value表示游戏对象的激活状态
public void SetActive(bool value);

//获取游戏对象的组件，参数type表示组件类型
//重载1：一个该类型的组件（深度优先）。
public Component GetComponent(Type type);
//重载2：获得所有该类型的组件（根据深度优先顺序依次编号）
public Component[] GetComponents(Type type);

GameObject

GameObject公有函数：

//获取游戏对象、父对象或者子对象中的的相应组件，参数type表示组件类型。
public Component GetComponentInChildren(Type type);
public Component[] GetComponentsInChildren(Type type);
public Component GetComponentInParent(Type type);
public Component[] GetComponentsInParent(Type type);

//注意：
//调用上面函数获取组件时，也会寻找游戏对象自身包含的相应组件。

Transform

Transfrom类包含游戏对象的位置、旋转、缩放比例等属性，该类用于控制游戏对象的Transform属性。

Transform公有字段：

```
public Vector3 forward;          //在transform坐标系下的正z方向。
public Vector3 right;            //在transform坐标系下的正x方向。
public Vector3 up;               //在transform坐标系下的正y方向。
public Vector3 position;         //该transform在世界坐标系下的位置。
public Vector3 localPosition;    //该transform相对其父对象的局部位置。
public Vector3 rotation;         //该transform在世界坐标系下的朝向。
public Vector3 localRotation;    //该transform相对其父对象的局部朝向。
public Vector3 scale;            //该transform在世界坐标下的缩放比例。
public Vector3 localScale;       //该transform相对其父对象的局部缩放比例。
public Transform parent;         //该transform的父对象。
public int childCount;           //该transform的子对象个数。
```

Transform

Transform公有函数：

//获得子对象的tranform，index:子对象索引。
public Transform GetChild(int index);

//设置父对象，parent:父对象， worldPositionStays:是否保持世界坐标不变：
public void SetParent(Transform parent, bool worldPositionStays)

//使游戏对象更改其transform属性以面朝目标物，target:目标物。
public void LookAt(Transform target);

//使游戏对象以自身为轴旋转（自转），eulerAngles表示旋转向量。
public void Rotate(Vector3 eulerAngles);

//使游戏对象绕着某个旋转中心点旋转（公转）。
// point:旋转中心点，axis:旋转轴，angle:旋转角度。
public void RotateAround(Vector3 point, Vector3 axis, float angle)

//使游戏对象以自身为参照系进行位移，translation:位移向量。
public void Translate(Vector3 translation);

Vector3

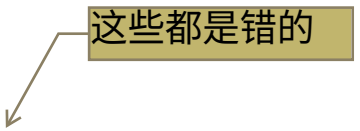
Vector3类用于三维向量的设置与计算。

Vector3公有字段：

public float x; public float y; public float z;　　//三维向量在x,y,z方向上的分量。
public float magnitude;　　//返回三维向量的模（向量的大小）。
public Vector3 normalized;　　//返回三维向量的标准化向量。

Vector3静态字段：

public static Vector3 forward：　　//表示三维向量(1,0,0)。
public static Vector3 back：　　//表示三维向量(-1,0,0)。
public static Vector3 right：　　//表示三维向量(0,1,0)。
public static Vector3 left：　　//表示三维向量(0,-1,0)。
public static Vector3 up：　　//表示三维向量(0,0,1)。
public static Vector3 down：　　//表示三维向量(0,0,-1)。
public static Vector3 zero：　　//表示三维向量(0,0,0)。
public static Vector3 one：　　//表示三维向量(1,1,1)。



Vector3

Vector3公有函数：

//设置三维向量在x,y,z方向上的分量。
public void Set(float new_x, float new_y, float new_z);

//将该三维向量标准化。
public void Normalize();

Vector3静态函数：

//计算两个三维向量（坐标点）的距离。
public static float Distance(Vector3 a, Vector3 b);

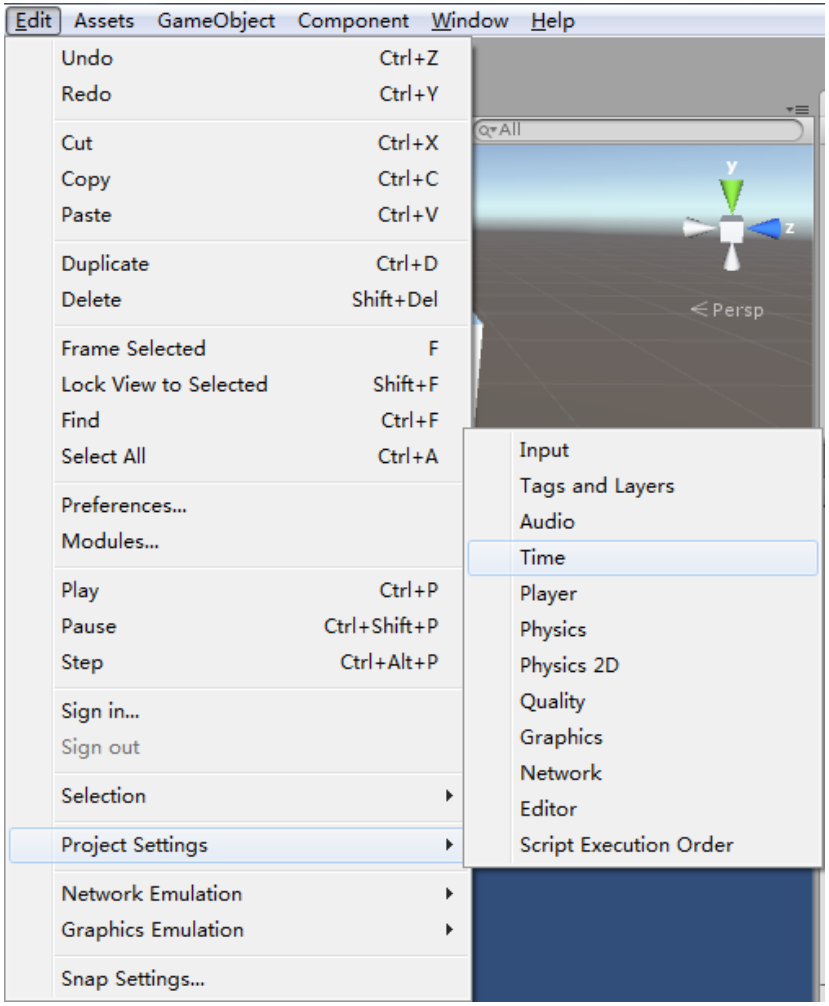
Time

Time类用于管理与游戏时间相关的属性。

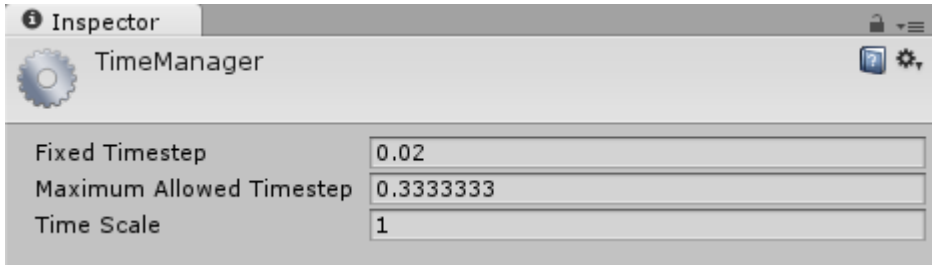
Time静态字段：

```
//上一帧（Update）所耗费的时间（只读）。
public static float deltaTime;

//上一次固定刷新（FixedUpdate）所耗费的时间（只读）。
public static float fixedDeltaTime;

//从游戏开始运行到调用此函数所经历的时间（只读）。
public static float time;

//时间缩放比例，现实中经过1秒时游戏经过的时间，用于放慢、加快游戏速度。
//当timeScale=1时表示与现实时间同步
//当timeScale<1时表示放慢游戏速度
//当timeScale>1时表示加快游戏速度。
public static float timescale;
```

Time



我们可以在**TimeManager**（Edit→Project Settings→Time）中修改固定刷新时间（Fixed Timestep）与时间缩放比例（Time Scale）。



Input

Input用于接收用户的输入信息。

Input静态函数：

// 鼠标输入函数：当鼠标发生相应动作时返回true，否则返回false。
// 参数button用于区分鼠标键：0表示左键、1表示右键、2表示中键。
public static bool GetMouseButton(int button);
public static bool GetMouseButtonDown(int button);
public static bool GetMouseButtonUp(int button);

//上述函数中：
// GetMouseButton在鼠标键保持持续按下时不断返回true；
// GetMouseButtonDown在鼠标键按下的动作发生时（一瞬间的动作）才返回true；
// GetMouseButtonUp在鼠标键抬起的动作发生时（一瞬间的动作）才返回true；

Input

Input静态函数：

```
//键输入函数：当键按键发生相应动作时返回true，否则返回false。
// 参数name表示InputManager 中设置的对应按键，如Input.GetKey(”Fire1”)
public static bool GetKey(string name);
public static bool GetKeyDown(string name);
public static bool GetKeyUp(string name);

//键输入函数：当键按键发生相应动作时返回true，否则返回false。
//参数key用于指定键盘上相应的键名，如Input.GetKey(KeyCode.W);
public static bool GetKey(KeyCode key);
public static bool GetKeyDown(KeyCode key);
public static bool GetKeyUp(KeyCode key);
```

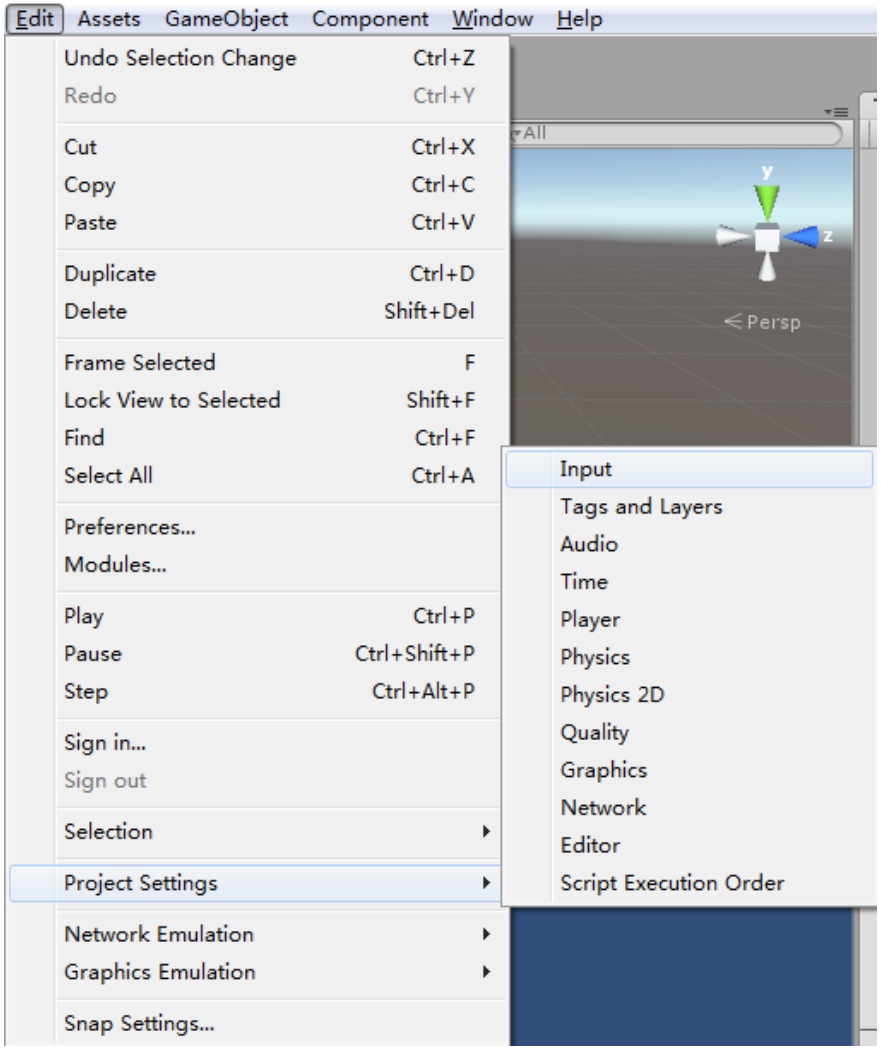
Input

Input静态函数：

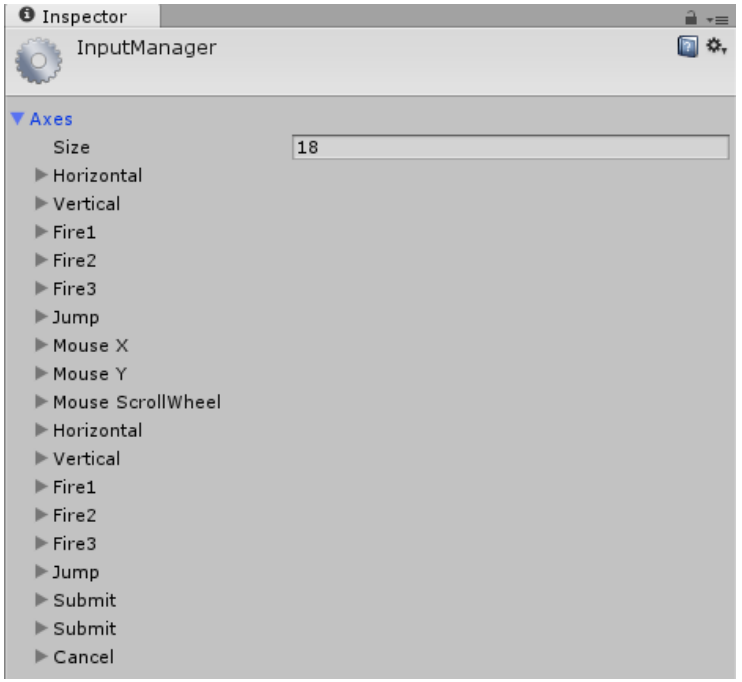
//方向轴输入函数：获取对应方向轴的按键的输入时返回true，否则返回false。
public static float GetAxis(string axisName);

/*
上述函数中：
参数axisName表示虚拟轴的名称，包括”Horizontal”、”Vertical”、”Mouse X”、”Mouse Y”。
我们使用”Horizontal”（水平）与”Vertical”（垂直）表示键盘或手柄在水平与垂直虚拟轴的输入，返回值的绝对值随着虚拟轴的持续输入将持续增加，返回值的范围为-1~1之间。
我们使用”Mouse X”与”Mouse Y”表示鼠标在水平与垂直虚拟轴的输入，返回值的绝对值大小表示鼠标移动的快慢。与键盘或手柄的虚拟轴不同，其返回值的范围不是-1~1。
*/

Input



我们可以在**InputManager**（Edit→Project Settings→Input）中修改按键名称与具体按键的对应关系。



Mathf

Mathf类表示数学类，包含一些常用的数学函数，用于处理数学上的运算。

Mathf静态字段：

```
public static float PI;          //表示π。
public static float Deg2Rad;     //角度变弧度所需要乘的系数，大小为π/180。
public static float Rad2Deg;     //弧度变角度所需要乘的系数，大小为180/π。
```

Mathf静态函数：

```
//三角函数与反三角函数，参数f表示弧度值。
public static float sin(float f);
public static float cos(float f);
public static float tan(float f);
public static float Asin(float f);
public static float Acos(float f);
public static float Atan(float f);
```

Mathf

Mathf静态字段：

```
//取绝对值
public static float Abs(float f);

//四舍五入
public static float Round(float f);

// 获取最大与最小值，
// 参数a、b表示需要比较大小的的两个数
// 参数values表示需要比较大小的数组。
public static float Max(float a, float b);
public static float Max(params float[] values);
public static float Min(float a, float b);
public static float Min(params float[] values);
```

Mathf

Mathf静态函数：

```
// 线性插值函数。参数a,b表示线性两端的值，t表示插值位置。
// 当t<0时，返回值为a；当t>1时，返回值为b；
// 当0<t<1时，返回值的计算方式为a+(b-a)/t。
public static float Lerp(float a, float b, float t);

// 限制范围函数，将目标值限定在min与max之间。
// 参数min,max分别表示范围内的最小最大值。
// 若value<min，则返回值为min；
// 若value>max，则返回值为max；
// 否则，返回值为value本身
public static float Clamp(float value, float min, float max);
```

Random

Random类提供了随机生成数值的功能。

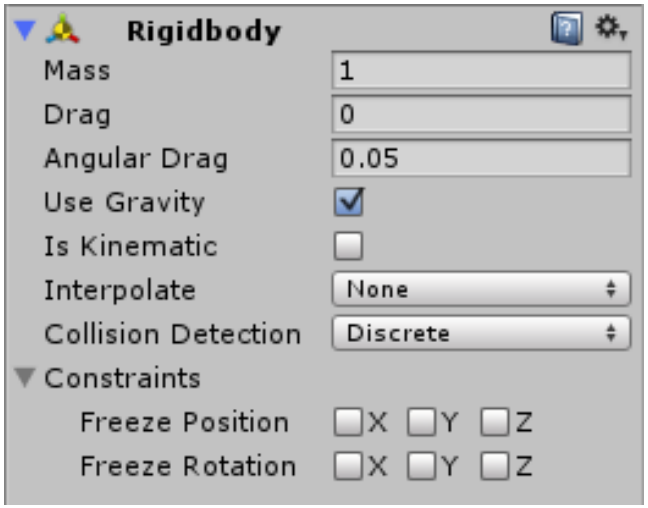
Random静态函数：

//在某一区间随机产生一个数, min表示区间下限， max表示区间上限
// Range函数随机产生[min,max]闭区间内的某个值。
public static float Range(float min, float max);

Unity其它功能类

- Rigidbody
- Collider
- Physics
- Light
- Camera
- AudioSource

Rigidbody



Rigidbody公有字段：

public float mass：设置刚体的质量属性。
public float drag：设置刚体的阻力属性。
public float angularDrag：设置刚体的角阻力属性。
public bool useGravity：设置刚体是否应用重力。
public bool isKinematic：设置刚体的is Kinematic属性。
public RigidbodyConstraints constraints：设置刚体的约束属性。

运动的，动态的

Rigidbody公有函数：

```
//刚体添加力
//参数force：表示添加力的向量；mode：所添加力的模式；
//ForceMode枚举（四种可选择的力模式）：
//      ForceMode.Force，持续作用力，受物体本身质量影响；
//      ForceMode.Acceleration，持续加速度，不受物体本身质量影响；
//      ForceMode.Impulse，瞬间作用力，受物体本身质量影响；
//      ForceMode.VelocityChange，瞬间速度变化，不受物体本身质量影响。
public void AddForce(Vector3 force, ForceMode mode = ForceMode.Force);
```

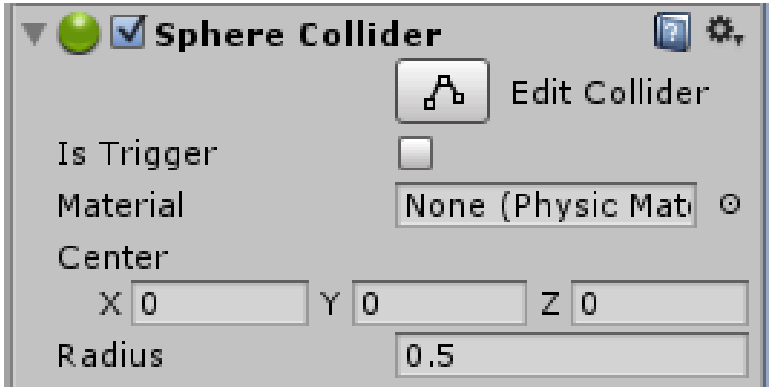
Collider

Collider公有字段：

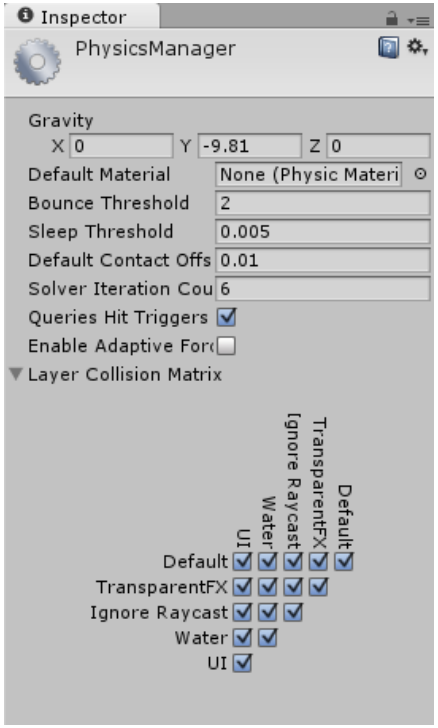
public Rigidbody attachedRigidbody：返回该碰撞体所连接的刚体，如无则该值为null。
public bool enabled：设置collider是否可用。
public bool isTrigger：设置collider是否为触发器（Trigger）。
public PhysicMaterial material：设置碰撞体组件的物理材质属性。

Collider生命周期函数：

OnCollisionEnter(Collision c)
OnCollisionStay(Collision c)
OnCollisionExit(Collision c)
OnTriggerEnter(Collider c)
OnTriggerStay(Collider c)
OnTriggerExit(Collider c)



Physics



Physics静态字段：

public static Vector3 gravity：设置场景中重力的大小与方向。
public static float bounceThreshold：设置两个对象发生碰撞时的最小相对速度。

Physics静态函数：

//发射物理射线
//参数ray表示射线信息，由射线起点origin （Vector3）与方向direction（Vector3）组成；
//参数hitInfo返回值包含了被击中碰撞体的信息，通过hitInfo.collider来获取；
//参数maxDistance表示射线的长度，若不传入该参数，则默认射线长度为无限长。
public static bool Raycast(Ray ray, out RaycastHit hitInfo, float maxDistance = Mathf.Infinity);

FPS游戏重要参数

Light

Light公有字段：

public LightType type：设置光源的类型。
public Color color：设置光源的颜色。
public float intensity：设置光源的强度，范围为0~8，1表示正常亮度。
public float bounceIntensity：设置反射光的强度，范围为0~8，1表示正常亮度。

Camera

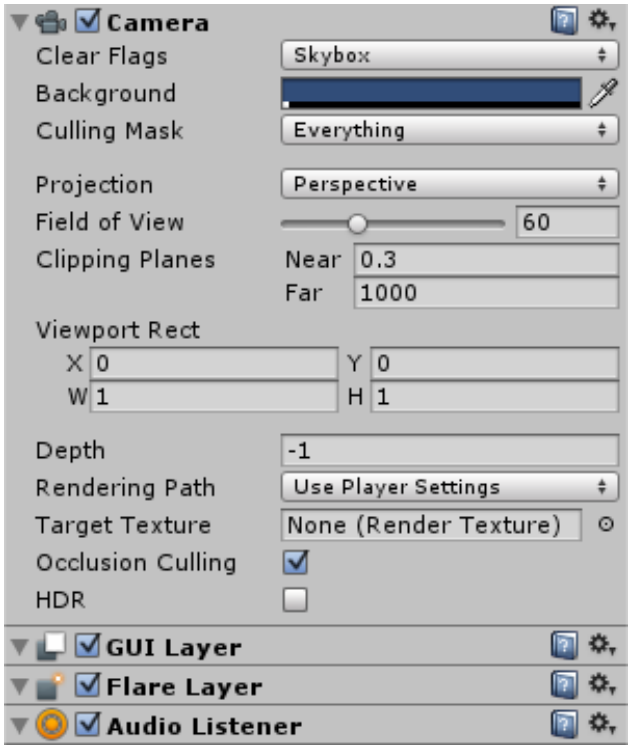
Camera静态字段：

public static Camera[] allCameras：获取包含场景中所有的摄像机。

Camera公有字段：

public CameraClearFlags clearFlags：设置摄像机清除标记。

public float depth：设置摄像机的深度值。



AudioSource

AudioSource公有字段：

public bool isPlaying：返回音频源是否在播放（只读）。
public bool loop：设置音频源是否循环播放。
public bool mute：设置音频源是否处于静默。
public float pitch：设置音频源的音调大小。
public bool playOnAwake：设置音频源是否自动播放。
public float volume：设置音频源的声音大小。

AudioSource公有函数：

```
public void Play();                                  //播放音频源
public void PlayDelayed(float delay);                //延迟delay秒后播放音频源
public void PlayOneShot(AudioClip clip);             //播放指定的音频剪辑clip
public void Pause();                                 //暂停音频源
public void UnPause();                               //继续播放音频源
public void Stop();                                  //停止播放音频源
```

AudioSource静态函数：

//在位置为position的地方播放音频剪辑clip。
public static void PlayClipAtPoint(AudioClip clip, Vector3 position)。