# 1.PhysicsSystem物理引擎全局设定详解

|  |
| --- |
|  |
|  |

## PhysicsSystem类的一些常用属性

|  |
| --- |
|  |
|  |

# 2.射线对象数据详解

|  |
| --- |
|  |

# 3.物理射线检测的使用步骤

|  |
| --- |
|  |
|  |
|  |
|  |

# 4.如何做3D屏幕拾取

|  |
| --- |
|  |

### 参考演练的代码

# 5.演练

## 1.新建一个项目，起名classTwentyone-ray,然后在assets文件夹里面新建3个文件夹：AssetsPackage，Scenes和Scripts，然后把主场景作为main保存到Scenes文件夹中

|  |
| --- |
|  |

## 2.在Scripts文件夹里面详新建一个GameMgr脚本,把脚本添加到摄像机节点中

|  |
| --- |
|  |

## 3.双击这个脚本用vscode打开,然后把多余的内容删除

|  |
| --- |
|  |

## 4. PhysicsSystem是cc模块提供的一个类,它有很多常用属性

|  |
| --- |
|  |
|  |

## 5.我们在场景中添加一个立方体,然后给他添加一个BoxCollider

|  |
| --- |
|  |

### 此时立方体才能够被射线检测到,没有碰撞器的话是检测不到的.

## 6.这个射线它是属于geometry的,我们可以在geometry上面点击右键->goto Definition来查看他的一些数据成员

|  |
| --- |
|  |

### 射线就是geometry.Ray,它有一个create方法来实例化对象

|  |
| --- |
|  |

### 还有一个克隆方法

|  |
| --- |
|  |

## 7.我们在场景在修改一些立方体的坐标,把它的z改为-10

|  |
| --- |
|  |

## 8.我们在脚本的start函数里面创建一个射线实例

|  |
| --- |
|  |

## 9.调用PhysicsSystem类的接口来进行射线检测,看看这条射线到底穿越了哪些物体, 有2种计算模式, PhysicsSystem.instance.raycastClosest方法,这种模式只要计算出一个物体就结束, PhysicsSystem.instance.raycast方法,这种模式会一直计算,直到所有的物体都计算完毕,这里使用计算一次的模式,

|  |
| --- |
|  |

### 运行程序,效果如下

|  |
| --- |
|  |

## 10.然后我们来传递mask参数

|  |
| --- |
|  |

### 此时运行程序,发现没有检测到物体,因为1<<1不能检测Default类型的物体,而这里的立方体就是Default类型

|  |
| --- |
|  |

### 其实这个参数的默认值是检测所有类型,所以,除非你有特殊需要,否则还是使用默认值好了

## 11.我们还是改回来了

|  |
| --- |
|  |

### 当然也可以使用|把几个类型连接起来

|  |
| --- |
|  |

## 12.我们可以把第一种方法注释了,我们来看看raycast方法,为了更好的观察结果,我们再新建一个立方体,需要添加碰撞器

|  |
| --- |
|  |

### 然后我们来写代码

|  |
| --- |
|  |

### 运行程序,效果如下

|  |
| --- |
|  |

# 下面,我们来学习屏幕拾取,也就是需要知道你点击的物体是什么

## 1.先把前面的代码注释了

|  |
| --- |
|  |

## 2.然后我们来监听屏幕触摸事件,先给编辑器绑定一个变量来获取摄像机实例(其实也可以不绑定,因为脚本就在摄像机上面)

|  |
| --- |
|  |

## 3.然后回到编辑器,把摄像机节点拖拽到我们的绑定的变量中的一栏中

|  |
| --- |
|  |

## 4.然后我们来监听TouchStart事件,需要传递一个回调函数

|  |
| --- |
|  |

### 运行程序,但我们点击一个物体的时候,控制台上就会输出这个物体的名字,点击屏幕的其他地方,就会显示没有碰到物体

|  |
| --- |
|  |