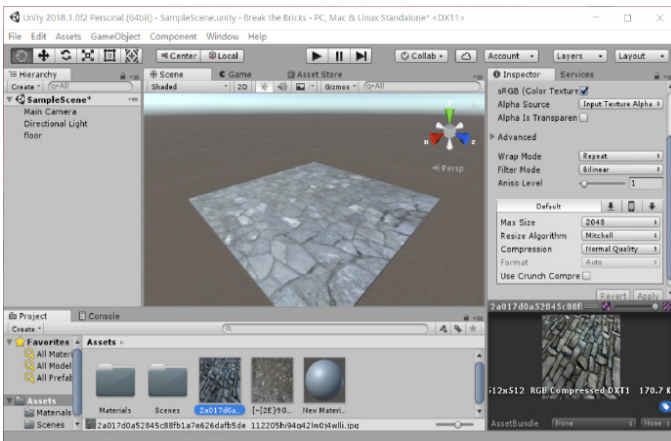






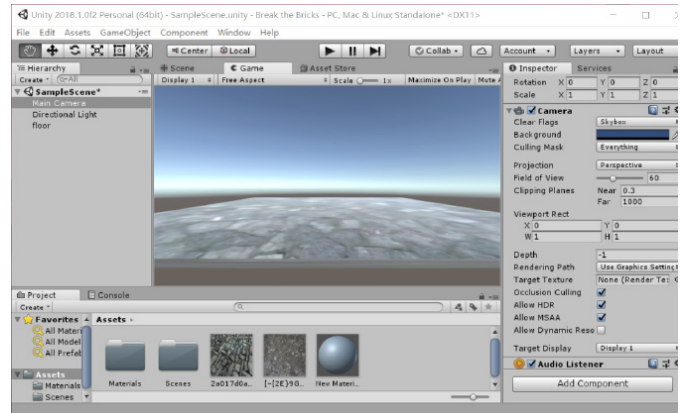
(P 3-7-14) 创建新场景

(2) 添加材质：白茫茫的地面不像地板，我们想让它有颜色或纹理，那么我们需要给物体添加材质。点击右下角的 create（如图所示）在里面选中 material，就会生成一个材质球，我们在右边的属性面板中选择 Albedo 用右面的取色器取你想要取的颜色，或如最后一张图所示，将一张地板的纹理图拖到下面的 assets 中再拖拽到“地面”上。



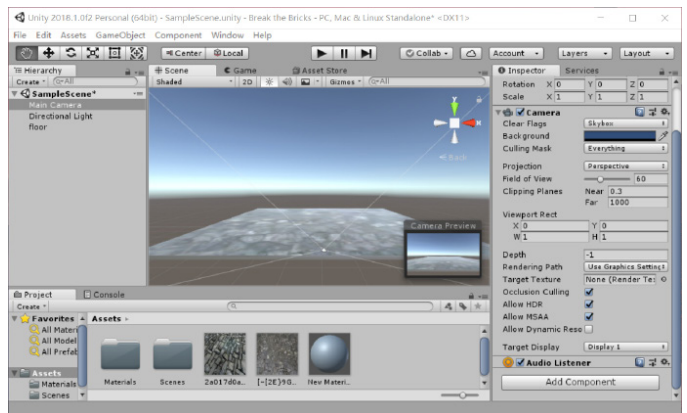
(P 3-7-15) 添加材质

(3) 移动主摄像机到合适的位置：选中场景中的主摄像机，通过调节     使得主摄像机呈现如图的效果：（点击场景区上方的 Game 查看游戏运行的画面，本例程中为主摄像机拍到的画面）。



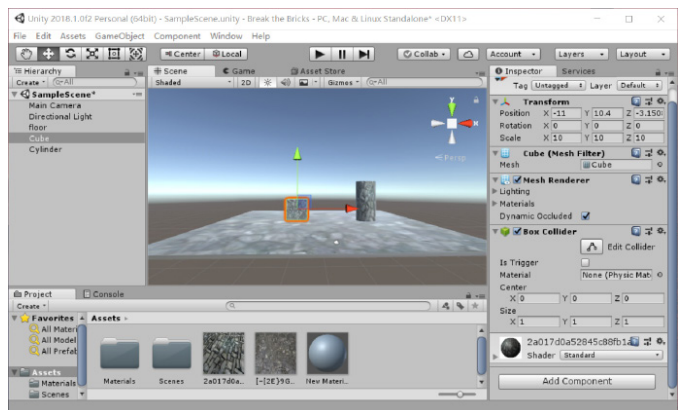
(P 3-7-16) 选合适得位置

把 scene 也调成这样的：



(P 3-7-17) 调场景

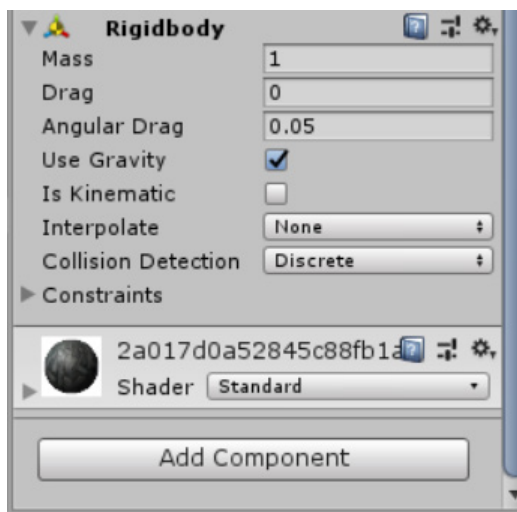
(4) 创建砖块：重复创建地面的操作，建立几个正方体或圆柱体，然后为它们添加纹理，例如：



(P 3-7-17) 创建砖块

这时的物块是没有物理属性的，它不具有质量和能量，炮弹碰它它没有反应，所以我们要给它添加刚体组件，点击最右下角的Add Component 选择Physics 然后选择Rigidbody，就给它添加了刚体组件（属性面板如图，Mass是它的质量），以后物体所需要的一些具有现实物体属性的组件都可以从Add Component 添加。

记得给物体改个名字，否则以后物体过多将无法标识。（注意：地面不能添加刚体组件！你可以添加并运行试试……）（刚体在大学物理上册第五章将接触到，也是一种物理模型，如果没学过的话，就简单地理解为，我给它赋予了重力和发生碰撞的能力）。

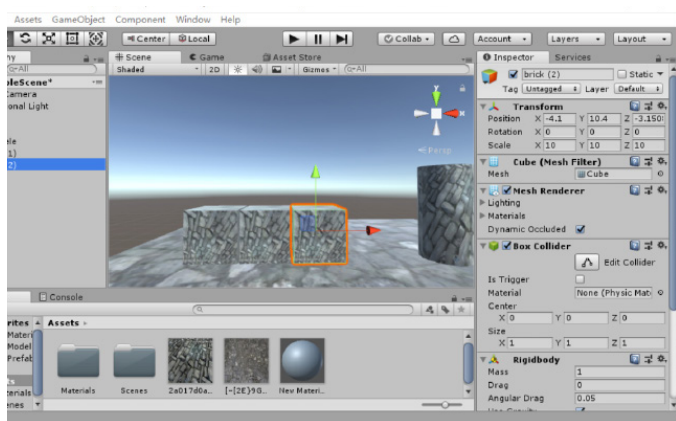


(P 3-7-18) 修改物体属性

(5) 复制物体：按 Ctrl+D 键在原地复制一个物体，然后用其中的

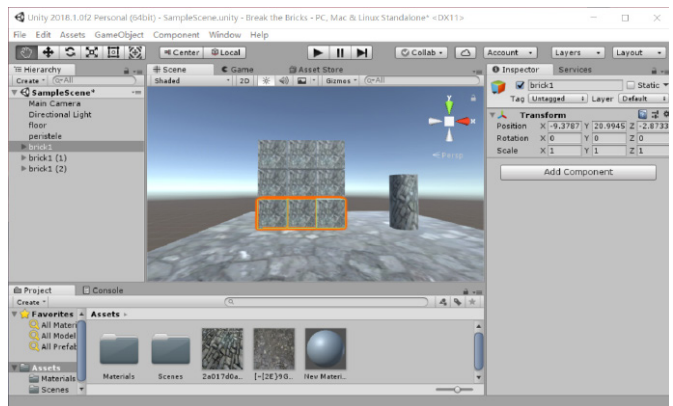


第二个将它们分开即



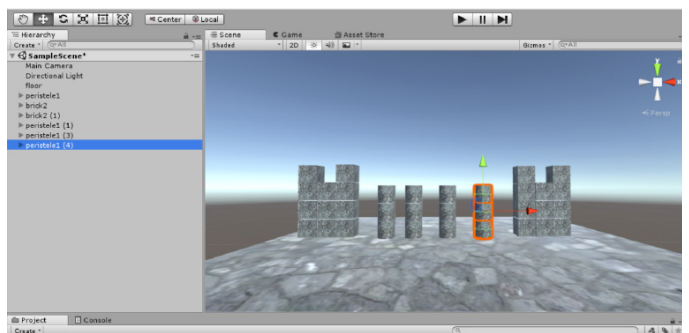
(P 3-7-18) 复制物体

如果要大批量复制，这样复制显然效率还是不够，我们这样做：在最上方点击GameObject，点第一个Create Empty 创建一个空物体改名为 brick1，在左侧的层级 hierarchy 面板中将所有的 brick 都拖拽到 brick1 中，将它们变成一个整体，然后批量复制，如图：



(P 3-7-19) 大量复制

对石柱 peristyle 也进行如上操作，并调整布局如下：



(P 3-7-20) 调整布局

6. 添加炮弹：

现在目标场景已经初步搭建好了，现在要为已有物体中添加炮弹。像创建物块那样创建一个球体Sphere 命名为 shell，并添加红色材质。为了让我们能随时调用这个物体，我们将其设为预置物体：将左侧层级 hierarchy 面板中的球拖拽到下方的项目面板中即可。

7. 添加脚本：

Unity3d 支持两种脚本，Javascript 和 c# 脚本，2018.1.0 及其以上的版本不支持 Js 脚本。这里以 c# 脚本为例。点击下册项目面板 Create 选 c# Script，就会出现一个脚本，我们将它命名为 shoot，双击它打开。没有 C# 编程基础的同学不要着急，这里只说一下各代码的作用，若同学有兴趣可以深学。

前三行是默认的不用管，// 后绿色的句子是注释不用管。第二段第一行默认创建了一个基类，用的是 shoot 的名字，包含了两个函数 void Start(场景一运行时即执行)和 void Update(在场景的每一帧被实时执行)，void 表示无特定返回值，不需要返回数据。

然后我们写一些指令，让主摄像机随键盘 W, S, A, D 的移动而移动，用户看到的就会是场景随 WASD 移动的效果。

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class shoot : MonoBehaviour
6 {
7     // Use this for initialization
8     void Start ()
9     {
10
11     }
12     int speed = 5;
13
14     // Update is called once per frame
15     void Update ()
16     {
17         float x = Input.GetAxis("horizontal") * Time.deltaTime * speed;
18         float z = Input.GetAxis("Vertical") * Time.deltaTime * speed;
19         transform.Translate(x, 0, z);
20         print(x);
21     }
22 }
```


(P 3-9-21) 代码注释

代码的意思分别为：

(1) int 是整数型，定义了速度变量，我们设速度的值为 5。

(2) float 为浮点型，定义了 x, z 两个变量，代表主摄像机在 x, z 两个方向，即前后两个方向上的位移。Input 表示输入。用户主要用键盘和鼠标输入，因此在 Unity 中定义了键盘鼠标输入时执行的操作，可以直接调用。这些我们在最上方 Edit->Projects->Settings->Input 中可以找到它的输入属性设置。GetAxis 是捕获轴，检

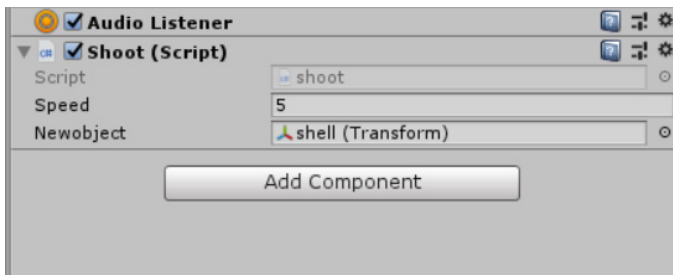
测键盘是否按下并返回一个数值，Horizontal 是横轴，默认由 AD 控制，Vertical 为纵轴，默认由 WS 控制，Time.deltaTime 是时间，speed 是速度，将这三个量相乘即可得到主摄像机在该方向上的位移。

(3) transform 组件是关于物体位置，角度，大小相关的组件，Translate 将 x, z 转换成其在 x, z 方向上的运动。写好之后我们将 C# 脚本拖拽到 Main Camera 上，并点击  中的运行，即可用 WASD 操控，如果嫌运动太慢可以适当将 C# 定义的 speed 速度值适当提高。下面是修改一些细节后的源码，注释在旁边：

```
1 public class shoot : MonoBehaviour
2 {
3     // Use this for initialization
4     void Start ()
5     {
6
7     }
8     public int speed = 30;
9     public Transform newobject;
10
11     // Update is called once per frame
12     void Update ()
13     {
14         float x = Input.GetAxis("horizontal") * Time.deltaTime * speed;
15         float z = Input.GetAxis("Vertical") * Time.deltaTime * speed;
16         transform.Translate(x, 0, z);
17
18         if(Input.GetButtonDown("Fire1"))
19         {
20             Transform n = Instantiate(newobject, transform.position, transform.rotation);
21             Vector3 fwd = transform.TransformDirection(Vector3.forward);
22             n.GetComponent<Rigidbody>().AddForce(fwd * 2800 * 5);
23         }
24
25         if(Input.GetKey(KeyCode.Q))
26         {
27             transform.Rotate(0, -25 * Time.deltaTime, 0, Space.Self);
28         }
29         if(Input.GetKey(KeyCode.E))
30         {
31             transform.Rotate(0, 25 * Time.deltaTime, 0, Space.Self);
32         }
33         if(Input.GetKey(KeyCode.Z))
34         {
35             transform.Rotate(-25 * Time.deltaTime, 0, 0, Space.Self);
36         }
37         if(Input.GetKey(KeyCode.C))
38         {
39             transform.Rotate(25 * Time.deltaTime, 0, 0, Space.Self);
40         }
41     }
42 }
```

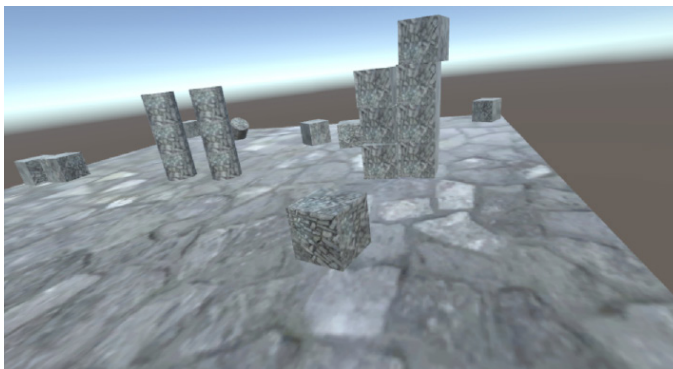
(P 3-7-22) 执行代码

然后把这个脚本添加在主摄像机上，这时属性面板中出现一个公有变量 **Newobject**，我们把刚添加的炮弹 **shell** 拖拽到这里边，然后运行。



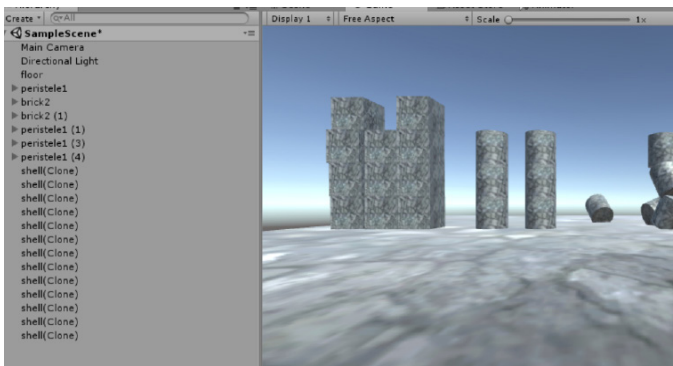
(P 3-7-23) 炮弹添加

效果如下：（发射炮弹后的废墟）



(P 3-7-24) 废墟

我们注意到发射很多炮弹后会生成很多炮弹，占用了许多内存空间，所以我们需要在炮弹发射后把它清理掉。



(P 3-7-25) 清理炮弹

因此，我们在 **shell** 球体上添加如下的脚本。

13 lines (11 sloc) | 239 Bytes

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class destory : MonoBehaviour
6 {
7     // Use this for initialization
8     void Start ()
9     {
10         Destroy(gameObject, 3);
11     }
12
13 }
```

(P 3-7-26) 执行代码

这样，游戏就初步制作成功了。

8. 设定规则：

只有场景是不好玩的，我们需要让场景与主角进行交互，或设定一些规则，让它变得有挑战性或奖励性，比如，可以规定主角在 15 秒内击落 10 个方块获胜，程序设计思想是：在所有砖块上添加一个脚本，实时监测它们的位置，当它们其中的一个在 y 方向上的坐标低于 -5 时记一个砖被消灭，影响一个标准位，然后访问 **GUIText** 让其显示“消灭数：xx”，编程让系统计时，当超过 15 秒时 $xx \geq 15$ 则 **GUIText** 显示“恭喜你赢了！”，当小于 15 秒时显示“再接再厉”。具体过程有点复杂，但代码量和上面那个差不多，这里不详述了。

9. 添加音乐：

依然是点击最右下角的 **Add Component**，添加 **Audio->Audio Source** 组件，在 **Clip** 中加入自己的音频源。（这个组件如果你加在主摄像机上，音乐就随着主角一起运动，如果加在物体上并加以设置，则越靠近物体音乐就会越响）



(P 2-7-27) 添加音乐

10. 添加按钮:

在进入,退出游戏,进入下一场景等操作时,添加按钮是必不可少的,这个就交给大家去探索啦!

11. 发布游戏:

点击最左上角的File->Build Settings->选择一个运行平台,这里选择PC/MAC/Linux平台,点Build即可发布,游戏发布成功后,就可以把它发给你的小伙伴们玩啦!

这个游戏比较简单,旨在简述游戏开发的

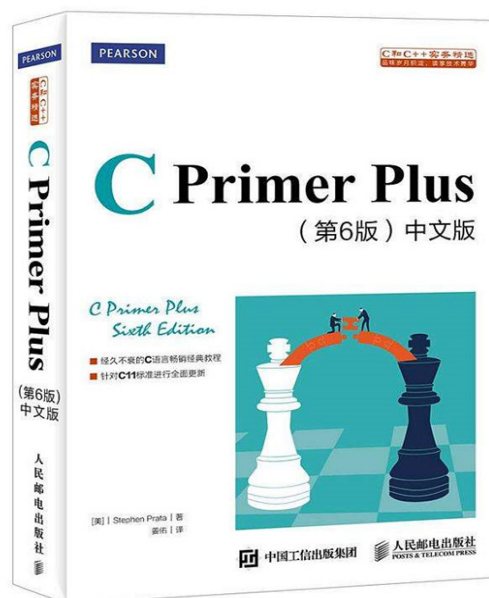
过程,Unity是一个很强大的引擎,大家如果有兴趣的话可以探索一下地形(GameObject->3dObject->Terrain)(可以轻松建立出一个沙盘模型),粒子特效,碰撞交互,第一/三人称控制器,动画播放等等。需要资源包的话可以到Unity官网上下载,也可以到网上去寻找,也可以去学习3dmax,从中导一些模型或材质的fbx文件给Unity。另外,Unity是VR引擎的一种,感兴趣的同学下来可以研究一下。

通院科协智网实验室推荐书目

一、C 语言学习

C 语言学习：《C Primer Plus》

书目介绍：全书共 17 章。第 1、2 章学习 C 语言编程所需的预备知识。第 3 到 15 章介绍了 C 语言的相关知识，包括数据类型、格式化输入输出、运算符、表达式、流程控制语句、函数、数组和指针、字符串操作、内存管理、位操作等等，知识内容都针对 C99 标准；另外，第 10 章强化了对指针的讨论，第 12 章引入了动态内存分配的概念，这些内容更加适合读者的需求。第 16 章和第 17 章讨论了 C 预处理器和 C 库函数、高级数据表示（数据结构）方面的内容。附录给出了各章后面复习题、编程练习的答案和丰富的 C 编程参考资料。



p(3-8-1) 《C Primer Plus》

二、单片机学习

单片机 (Microcontrollers) 是一种集成电路芯片，是采用超大规模集成电路技术把具有数据处理能力的中央处理器 CPU、随机存储器 RAM、只读存储器 ROM、多种 I/O 口和中断系统、定时器 / 计数器等功能（可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D 转换器等电路）集成到一块硅片上构成的一个小而完善的微型计算机系统，在工业控制领域广泛应用。从上世纪 80 年代，由当时的 4 位、8 位单片机，发展到现在的 300M 的高速单片机。

单片机诞生于 1971 年，经历了 SCM、MCU、SoC 三大阶段，早期的 SCM 单片机都是 8 位或 4 位的。随着工业控制领域要求的提高，开始出现了 16 位单片机，但因为性价比不理想并未得到很广泛的应用。90 年代后随着消费电子产品大发展，单片机技术得到了巨大提高。随着

INTEL i960 系列特别是后来的 ARM 系列的广泛应用，32 位单片机迅速取代 16 位单片机的高端地位，并且进入主流市场。

而传统的 8 位单片机的性能也得到了飞速提高，处理能力比起 80 年代提高了数百倍。目前，高端的 32 位 Soc 单片机主频已经超过 300MHz，性能直追 90 年代中期的专用处理器，而普通的型号出厂价格跌落至 1 美元，最高端的型号也只有 10 美元。

当代单片机系统已经不再只在裸机环境下开发和使用，大量专用的嵌入式操作系统被广泛应用在全系列的单片机上。而在作为掌上电脑和手机核心处理的高端单片机甚至可以直接使用专用的 Windows 和 Linux 操作系统。