

摘要

第一章 项目分析

- 1.1 游戏背景
- 1.2 游戏开发平台介绍

第二章 需求分析

- 2.1 游戏性能分析
- 2.2 功能分析
- 2.3 用户需求分析
- 2.4 游戏道具分析
- 2.5 玩家步骤分析-游戏流程以及流程图

第三章 游戏开发步骤

- 3.1 Unity编辑器快速概述
 - 3.1.1 工具栏介绍
 - I. 操作对象
 - II. 用鼠标导航
 - 3.1.2 场景模型导入
- 3.2 Scene中添加细菌
 - 3.2.1 为游戏创建关卡
 - 3.2.2 添加更多目标
 - 3.2.3 更改目标统计
 - 3.2.4 创建一个新目标
 - 3.2.5 添加目标动画和声音
 - 3.2.6 将目标变成预制件
- 3.3 关卡设计
 - 3.3.1 设计自己的关卡
 - 3.3.2 关闭音频小控件
 - 3.3.3 添加新房间和走廊
 - 3.3.4 添加上锁的门
 - 3.3.5 将钥匙与锁关联
- 3.4 武器修改
 - 3.4.1 创建新武器
 - 3.4.2 调整新武器预制件
 - 3.4.3 使玩家可以使用武器
- 3.5 游戏终点及整体逻辑
 - 3.5.1 创建一个LevelEnd
 - 3.5.2 添加触发器
- 3.6 开始UI界面及发布游戏
 - 3.6.1 开始UI界面
 - I. 创建 Button
 - II. 点击时场景切换

第四章 碰撞检测

- 4.1 碰撞检测
 - 1. Box Collider
 - 2. Capsule Collider
 - 3. Mesh Collider
 - 4. Wheel Collider

4.2 通关机制设计

第五章 问题及心得

- 5.1 问题及解决方案
 - 5.1.1 问题
 - 5.1.2 解决方案

- I. 基于unity3D场景切换所遇到问题及解决方法
- II. 素材的收集以及无法导入
- III. 物理组件和碰撞器的使用
- IV. 地图场景的搭建
- V. 门和钥匙Key - Value的对应

5.2 游戏开发心得

第六章 小组分工

第七章 游戏源代码

Scripts

- 7.1 游戏开始Start UI 场景切换
- 7.2 游戏音频
- 7.3 游戏中创建新场景
- 7.4 第一人称视角及特效
- 7.5 钥匙和锁
- 7.6 地图场景和主场景部分代码
- 7.7 MiniMap
- 7.8 武器
- 7.8 游戏主逻辑
- 7.9 游戏结束判断

摘要

游戏，作为大众化娱乐的方式之一，已经越来越受到各年龄段的人们的欢迎。而其中3D游戏也是技术含量比较高的一种。Unity 3D作为一种2D/3D游戏开发引擎，凭借其在音频，图像，物理等各种引擎上的强大功能，深受游戏开发者和爱好者的喜爱。

本文以 Unity3D 2019.2.10 为开发环境，设计并开发了一款单人射击类3D游戏——“健康保卫战”。本文旨在介绍Unity3D的开发流程和步骤，并逐步介绍3D游戏从前期策划，素材制作，到创建场景，编写脚本，调试代码，不断试验等，再到最后通过Unity3D平台生成一个可执行文件，完成一个完整的3D类游戏制作的全过程。

第一章 项目分析

1.1 游戏背景

人类健康深受病毒的威胁，一位勇士临危受命，带上胶囊手枪，不惧各种奇形怪状的病毒的阻挠，用勇气去对对抗敌人，用智慧去探索前方的道路。当然，也要时刻注意沿路的补给包。拿上枪，你就是人类最后的希望！

在未来，人们都生活在网络科技搭建的虚拟世界里，长期的不运动，全世界的人都是依靠各种药物维持自己的身体机能，每个人依靠生物药物改变身材改变相貌改变性格，忽然爆发了有史以来的最大的生化危机，生物专家斥巨资研究针对此次病毒的药物，可是这类药物无法口服无法注射，因为这类病毒在人类每个部位衍生楚各种各样不同的细菌，所以需要一位研究该病毒的生物学家服下缩小身体的药物进入病人身体，根据细菌衍生速度以及当前生长状态来实施处理。

在这危难之际，给所有的生物学家检查身体是否符合进入人体的体质，可是竟然无一人符合，因为生活在虚拟世界的人们身体机能已完全滞留，自身无法分泌促进生长的代谢的激素，这时有一位虽然生活在虚拟世界可是却从未放弃按时进食按时锻炼身体的一名医学生迪西勇敢站出来，愿意服下缩小药进入病人体内解决此次危机。

那么你们是否愿意进入虚拟世界同迪西一起，临危受命，带上胶囊手枪，不惧各种奇形怪状的病毒的阻挠，用勇气去对对抗敌人，用智慧去探索前方的道路，请加入我们，你就是人类最后的希望！

1.2 游戏开发平台介绍

本游戏开发环境为Unity3D，Unity 是一款跨平台的专业游戏引擎，可以使用它轻松的开发各种 2D 和 3D 游戏，然后部署到各种游戏平台上。当然也包括这些主流游戏平台：Windows、iOS、Android、Xbox 360、PS3。

第二章 需求分析

2.1 游戏性能分析

为了能流畅运行本射击游戏，需要您的个人计算机上至少有如下配置……

适用于各种PC端。（64位计算机和32位计算机）

2.2 功能分析

本游戏是3D射击类游戏，主要是由玩家的第一视角为主，配以地图辅助，射击游戏中出现的各种病毒形状的敌人，收集积分和钥匙道具，完成地图中所有路径的探索，即为通关。

本游戏是3D类游戏，主要是玩家的通过消灭病菌拿到钥匙，然后根据游戏提示或者尝试性分析找到每个门的钥匙，但此过程需要合理分析，稍有不慎可能又要重新返回，当开启最后一扇门进入场景后即为胜利。

该游戏利用 Unity3D 跨平台功能的突出、兼容性强的特点，结合 C# 开发一款坦克大战的游戏系统，画面好看，游戏方式简单，玩家易上手。

2.3 用户需求分析

本游戏简单上手，适合于不同年龄段的人群，但游戏菜单为英文模式，所以需要玩家有一定的英语阅读能力，英语四级及以上为佳。

本游戏简单上手，也易于操作，适合于不同年龄段的人群，画面充满活力，场景选择为人体的器官（如心脏，牙齿）以及动脉血管、毛细血管。更适用于有益智闯关游戏需求的朋友。

2.4 游戏道具分析

玩家：控制游戏中的射机枪，向游戏中的病毒发出攻击

病毒：不会攻击玩家，主要起到在被“射杀”后提供游戏的钥匙道具

钥匙：作为开启游戏路径中“门”的道具

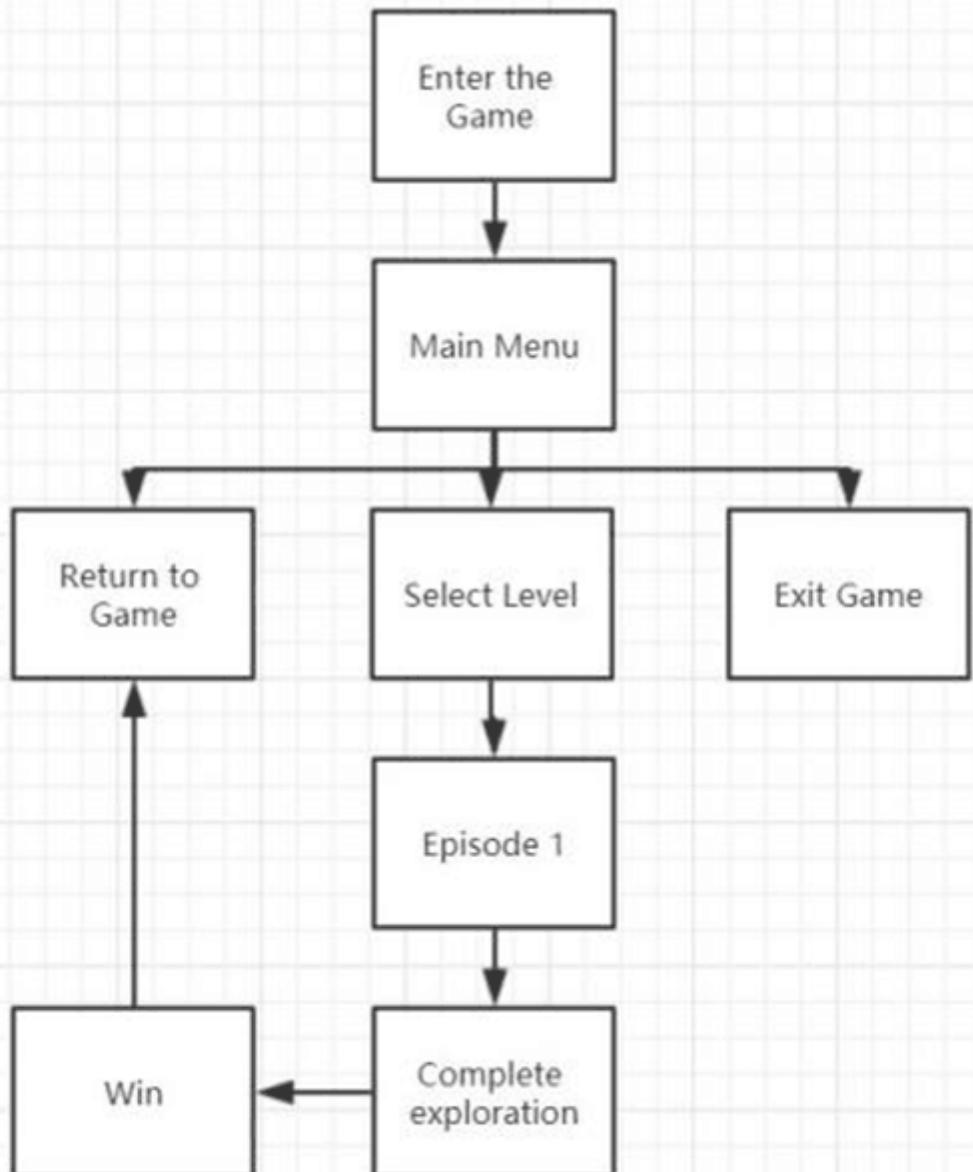
胶囊：“枪药”的“子弹”

1.玩家：使用“WASD”键分别控制游戏中人物，使用space键跳跃，鼠标左键发射子弹，鼠标滚轮控制武器的切换

2.血条生命值：场景内部有红色的血细胞可以增加人物血量

3.子弹：使用 R 键填装子弹

2.5 玩家步骤分析-游戏流程以及流程图

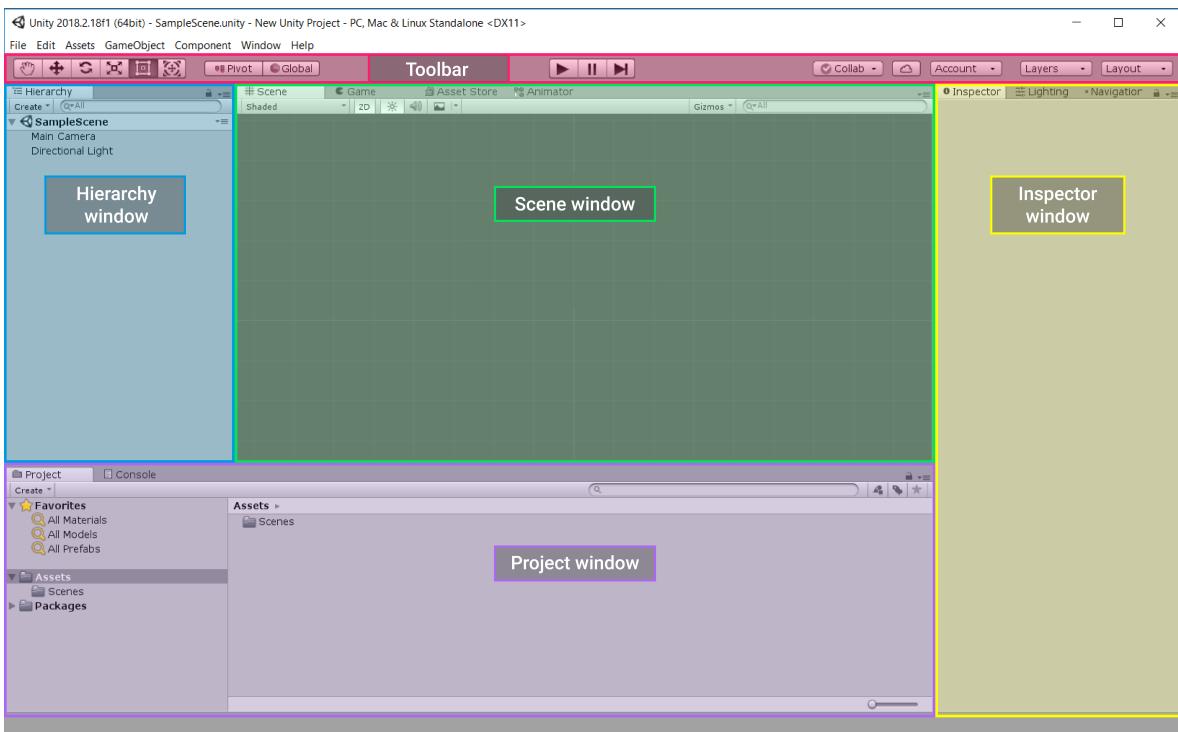


https://blog.csdn.net/weixin_43232955

第三章 游戏开发步骤

3.1 Unity编辑器快速概述

Unity编辑器的基本熟练



Unity编辑器有四个主要部分：

1.场景视图 在这里，您可以通过在3D空间中选择和移动游戏对象来编辑当前场景。在此套件中，游戏级别包含在一个场景中。

2.层次结构窗口 这是场景中所有GameObjects的列表。GameObjects包含将用于填充游戏角色、道具和风景。这些可以放置在父子层次结构中，这使您可以对对象进行分组-这意味着当父GameObject移动时，其所有子对象将同时移动。

3.检查器窗口 这将显示与当前所选GameObject相关的所有设置。在演练期间，您将更多地探索此窗口。

4.项目窗口 在这里，您可以管理项目**资产**。资产就是我们在项目中使用的所有媒体文件（例如，图像，3D模型和声音文件）。“项目”窗口的作用类似于文件浏览器，可用于在计算机上浏览和创建文件夹。当演练提示您在给定的文件路径中找到**资产**时，请使用此窗口。

提示：如果您的编辑器布局与上面的图像不匹配，请使用工具栏右上方的布局下拉菜单选择**默认**。

5.工具栏简介

工具栏包含一系列有用的工具按钮，可帮助您设计和测试游戏。



3.1.1 工具栏介绍

播放按钮



玩

“播放”用于测试当前在“层次结构”窗口中加载的“场景”，并允许您在编辑器中实时尝试游戏。

暂停

您可能已经猜到了，暂停可以让您暂停游戏窗口中的游戏。这可以帮助您发现在其他情况下不会看到的视觉问题或游戏性问题。

下一步

步骤用于逐帧浏览暂停的场景。当您在游戏世界中寻找实时变化时，这非常有用，这对于实时查看很有帮助。

I. 操作对象



这些工具可在“场景”视图中移动和操纵GameObject。您可以单击按钮将其激活，也可以使用快捷键。

手工具



快捷键：Q

您可以使用此工具在窗口中四处移动场景。您也可以使用鼠标中键单击来访问该工具。

移动工具



快捷方式：W

该工具使您可以选择项目并分别移动它们。

旋转工具



快捷键：E

选择项目并使用此工具旋转它们。

比例工具



快捷键：R

您猜对了-这是用于缩放游戏对象的工具。

矩形转换工具



捷径：T

这个工具可以做很多事情。从本质上讲，它将移动，缩放和旋转组合为一个专用于2D和UI的工具。

旋转，移动或缩放



捷径: Y

同样，该工具可以完成很多事情。它还使您能够移动，旋转或缩放GameObject，但是它更适合3D。

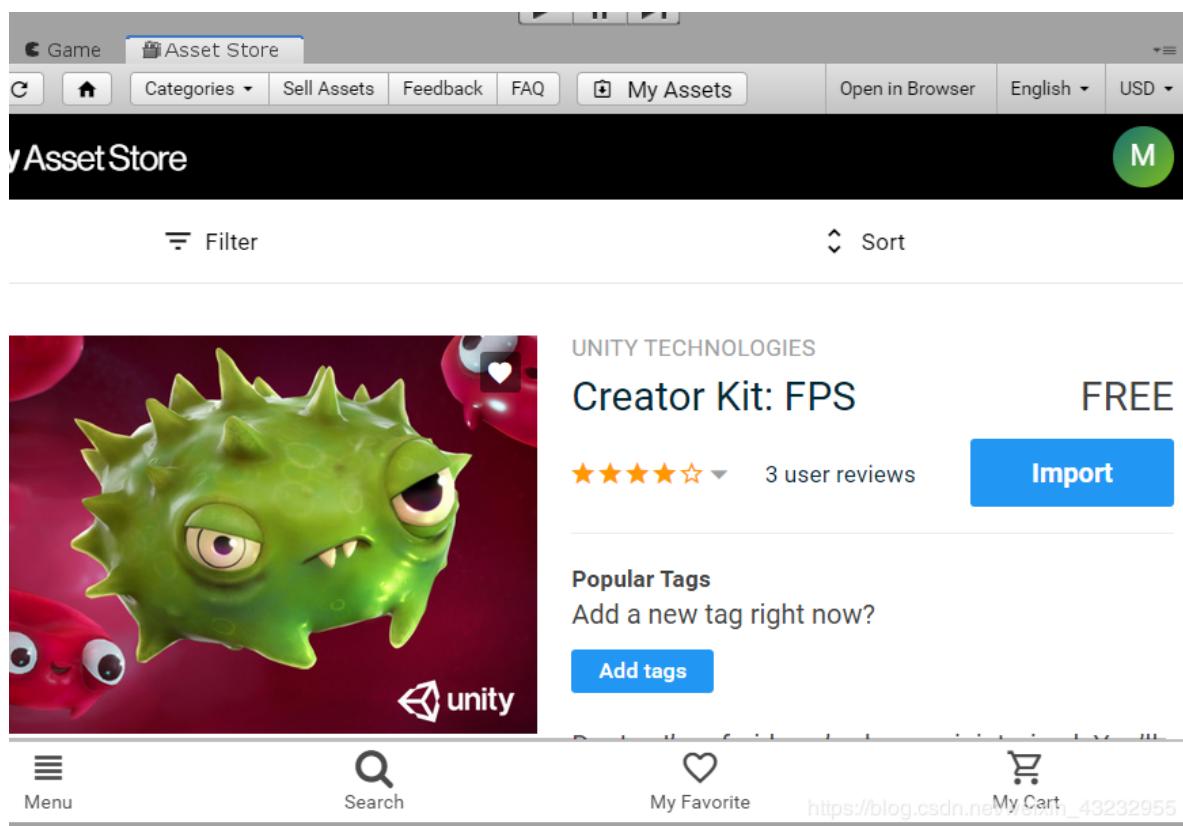
II. 用鼠标导航

在“场景”视图中时，还可以：

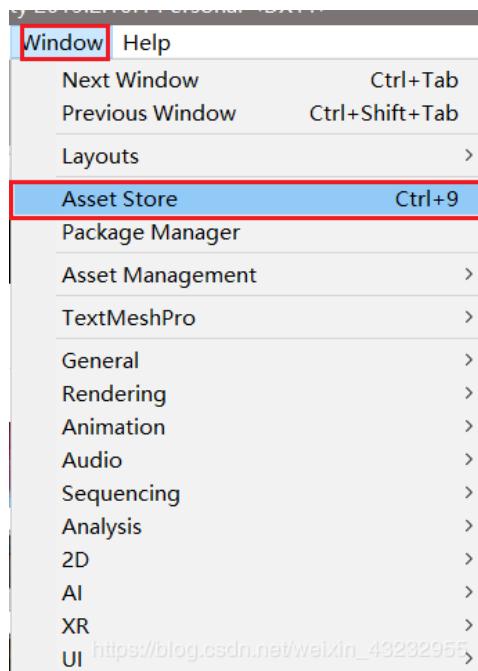
- 左键单击以在场景中选择您的GameObject。
- 单击鼠标中键并拖动以使用手动工具移动“场景”视图的相机。
- 右键单击并拖动以使用“飞越”模式（手动工具的一种）来旋转“场景”视图的相机。在执行此操作时，您还可以使用A和D左右移动相机，使用W和S前后移动相机，使用Q和E上下移动相机。

3.1.2 场景模型导入

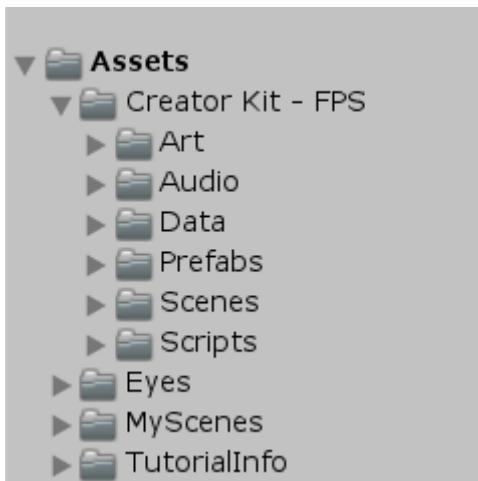
首先，导入素材场景：



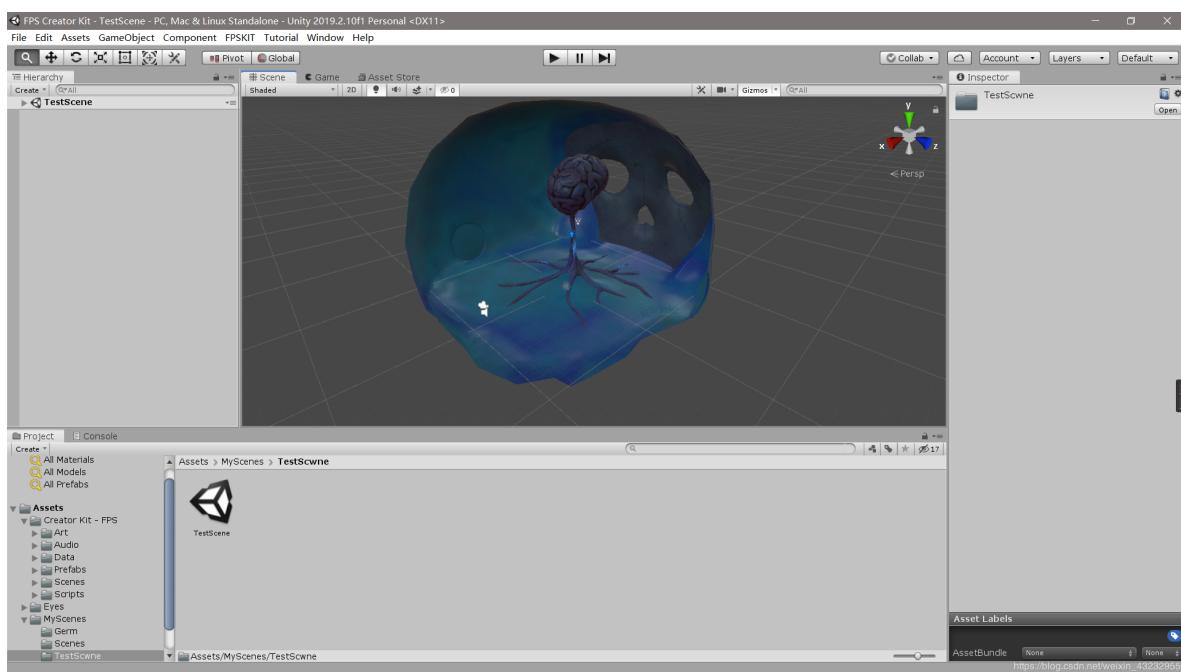
1. 在“Windows”窗口中，导航到“Asset Store”选项



2. 寻找合适的素材并导入



打开场景后，就可以看到模型了。选择工具栏中的“播放”按钮开始。



此时，仅仅是一个空场景，而我们最终想要实现的目标是、

I. 游戏角色动作目标

游戏能够

- 按下键盘上的 WASD 键移动
- 按下空格键即可跳转
- 按住 Shift 键运行
- 用鼠标瞄准武器
- 瞄准后，通过单击鼠标按钮用药消灭细菌
- 消灭细菌会给你点药，并消耗药弹。您可以在屏幕的左上角看到您的得分，并在右下角看到武器的剩余药量
- 当你完成游戏测试员，按逃逸 键盘上，让您的鼠标光标回来。

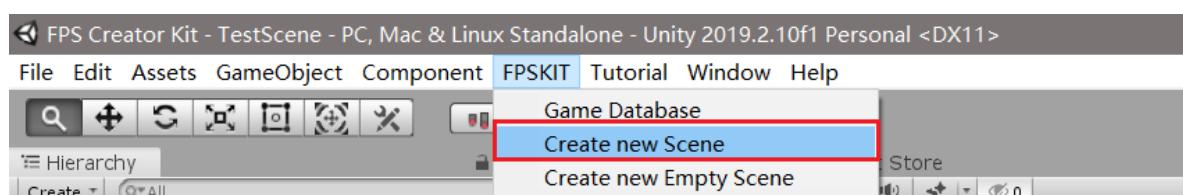


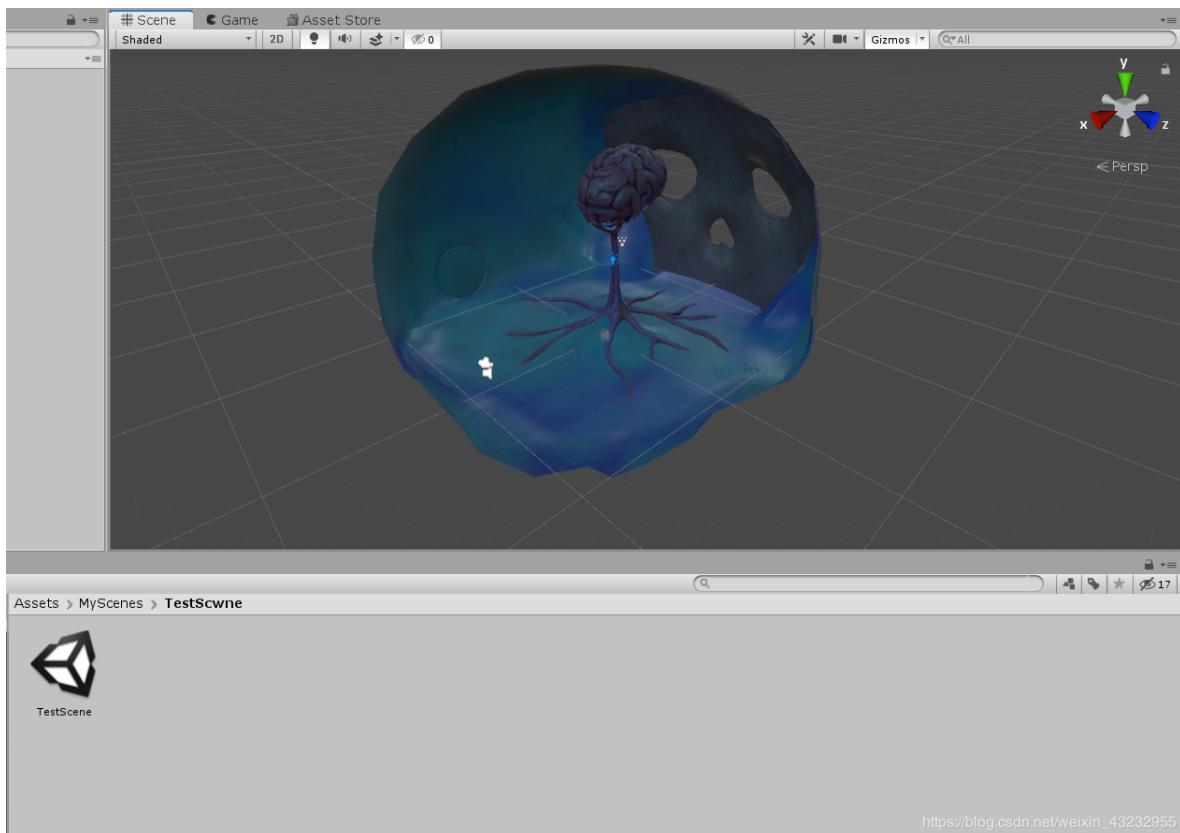
3.2 Scene中添加细菌

3.2.1 为游戏创建关卡

在游戏模型中创建了一个模板，其中包含一个房间，制作自己的副本：

- 从顶部菜单栏中，转到 FPSKIT -> 创建新场景



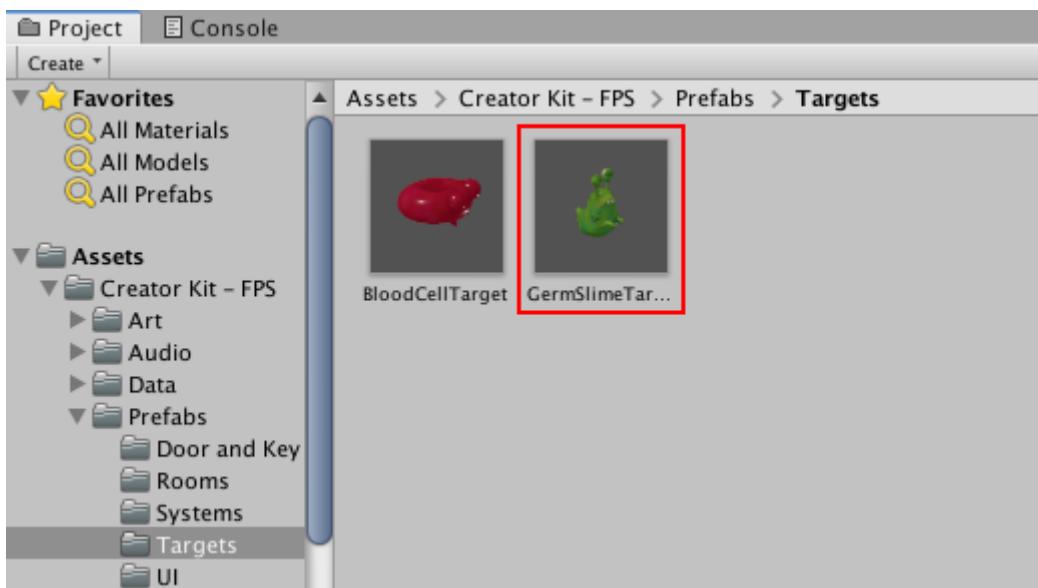


2. 为新的场景指定唯一的名称，然后单击“**保存**”。这将在**Assets -> MyScenes**中 创建新的Scene，并打开它

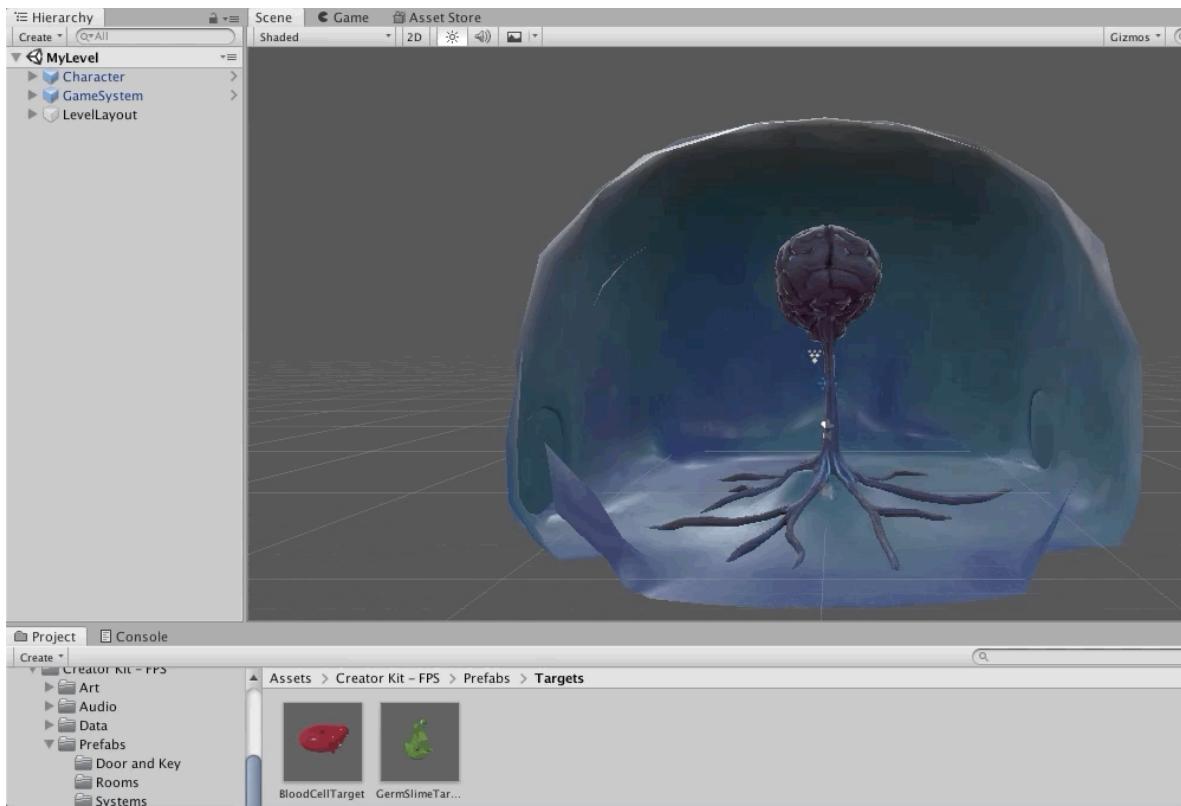
3.2.2 添加更多目标

从向房间添加目标开始：

1. 在“**项目**”窗口中，转到“**资产/创建者套件FPS / Prefabs / Targets**”。选择**GermSlimeTarget** 预制件。



2. 将**GermSlimeTarget** 预制件从“项目”窗口拖动到“场景”视图中。



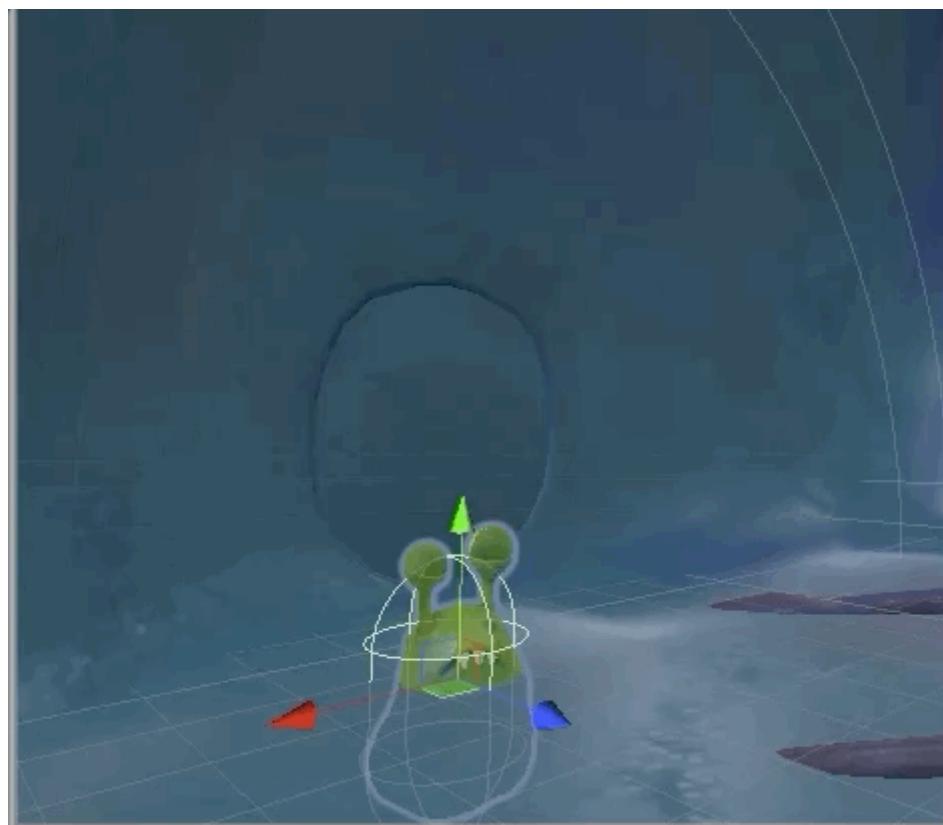
现在，在左侧的“层次结构”窗口中看到GermSlime。已成功在场景中放置了一个新的GameObject。

3. 现在，将GermSlime移到新位置。首先，在工具栏中选择“**移动工具**”。



能够看到覆盖在新目标上的三个箭头（蓝色，绿色和红色）。

4. 首先，单击**绿色箭头**并向上拖动以更改GermSlimeTarget的垂直位置。这会将其抬离房间的地板。



5. 单击并拖动**红色和蓝色箭头**以更改GermSlimeTarget的水平位置。将其放在房间中的任何位置。



还可以选择三个移动箭头中间的立方体，以一次在多个方向上移动GameObject。

6. 将GermSlimeTarget放置在所需位置后，按Ctrl + S 保存场景。
7. 选择“**玩**”以再次测试游戏，然后尝试销毁新细菌。您可以根据需要向游戏添加尽可能多的新目标。

3.2.3 更改目标统计

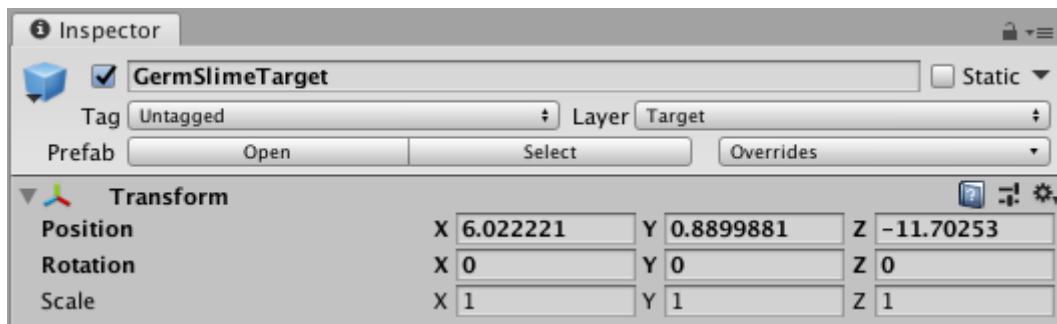
该游戏中的Prefab靶标具有简单的统计信息，可用来衡量它们的**健康状况**以及被药击中时获得的**积分值**。可以更改这些统计信息，使细菌更容易或更难以治愈。

调整目标的统计信息：

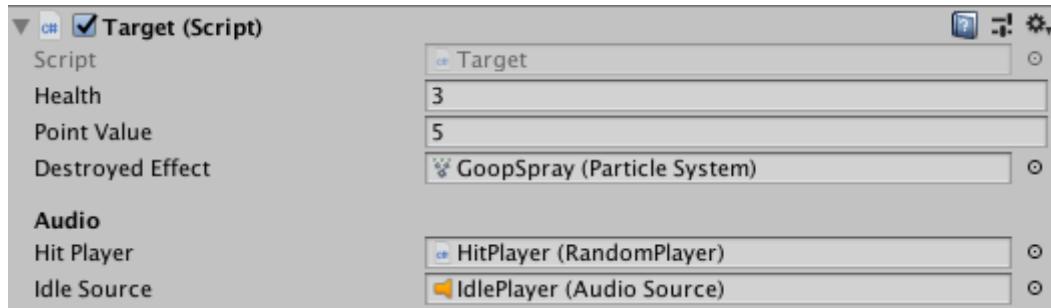
1. 在“场景”视图中，单击GermSlimeTarget。



“**检查器**”窗口现在将显示目标的设置。“**检查器**”窗口显示了连接到GameObject的所有**组件**。组件用于在Unity中向GameObjects添加不同的功能。



2. 更改目标的健康状况和积分值。



增加其**健康状况**将使细菌更难被破坏，而减少其**健康状况**将使其变得更容易。销毁细菌时，设置的**点值**将显示在“场景”视图的左上角。

3. 按 **Ctrl + S** 保存游戏。现在播放以测试更改。

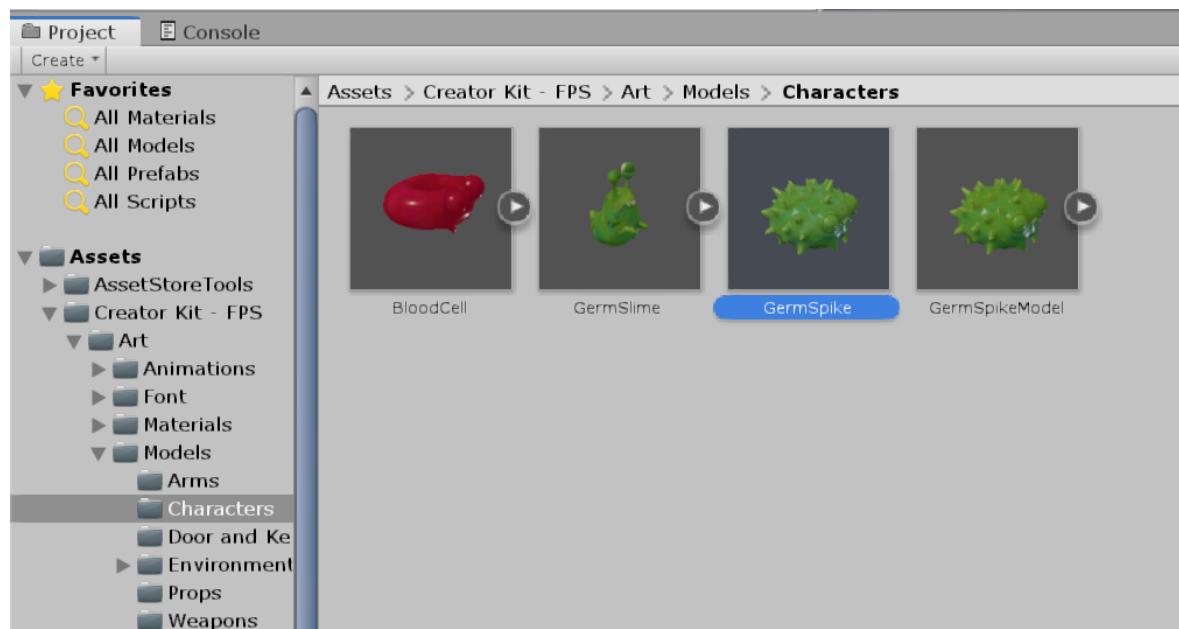
游戏中已经存在的红细胞也可以被医生的药物作为目标。可以选择一个红细胞，并以完全相同的方式找出其“健康”和“点值”。

已经为游戏准备好了GermSlimes，现在就可以从头开始创建新目标了！

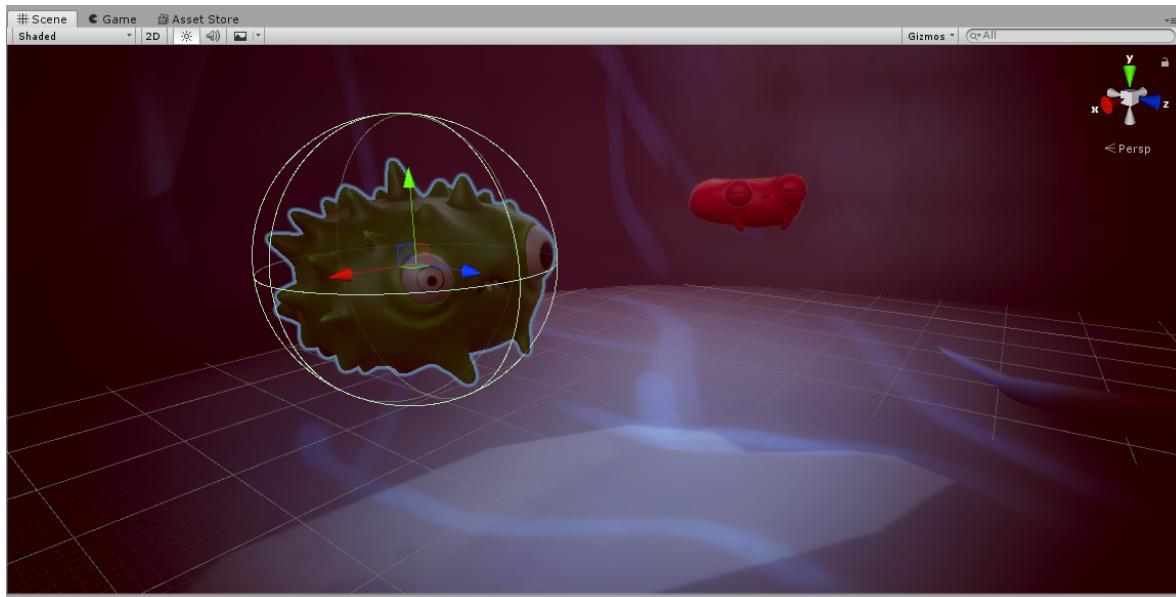
3.2.4 创建一个新目标

为您的游戏创建一个新的细菌目标：

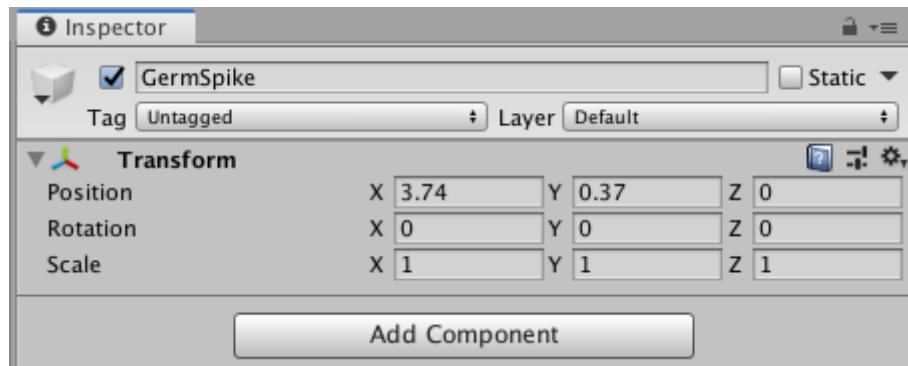
1. 在**Project窗口中**，转到 *Assets / Creator Kit-FPS / Art / Models / Characters*，然后选择**GermSpike**资产。



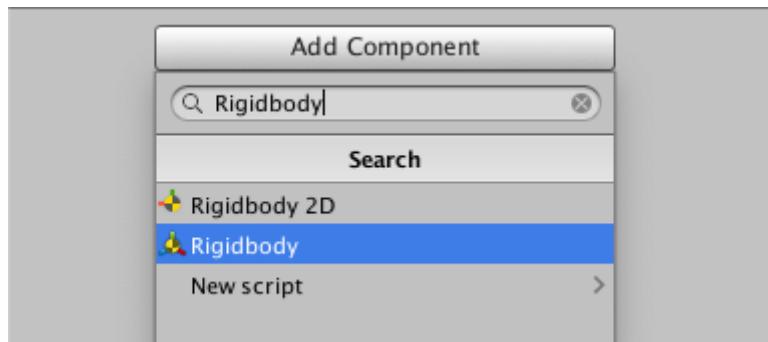
2. 将模型拖放到“场景”视图中。



3. 确保已选择对象，然后查看“检查器”窗口。您现在应该只看到一个Transform组件。



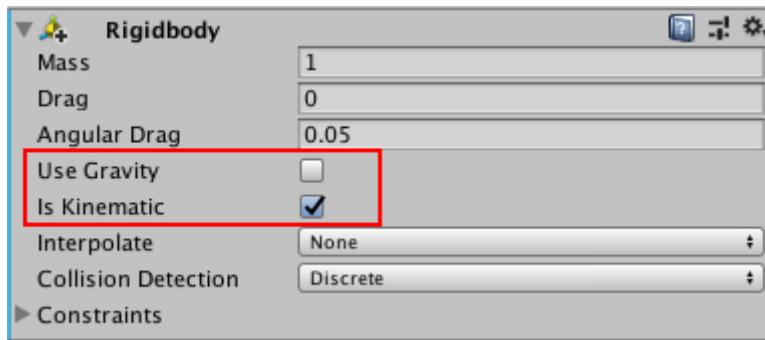
4. 单击添加组件按钮，然后开始在搜索栏中输入“Rigidbody”。



单击Rigidbody组件将其添加到GermSpine。

5. 在“检查器”窗口中找到“刚体”组件，然后：

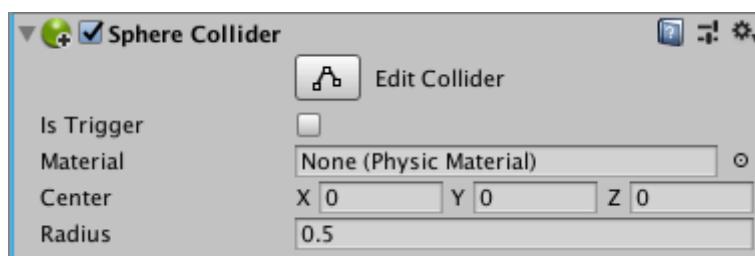
- 禁用**使用重力**复选框
- 启用**是运动学**复选框



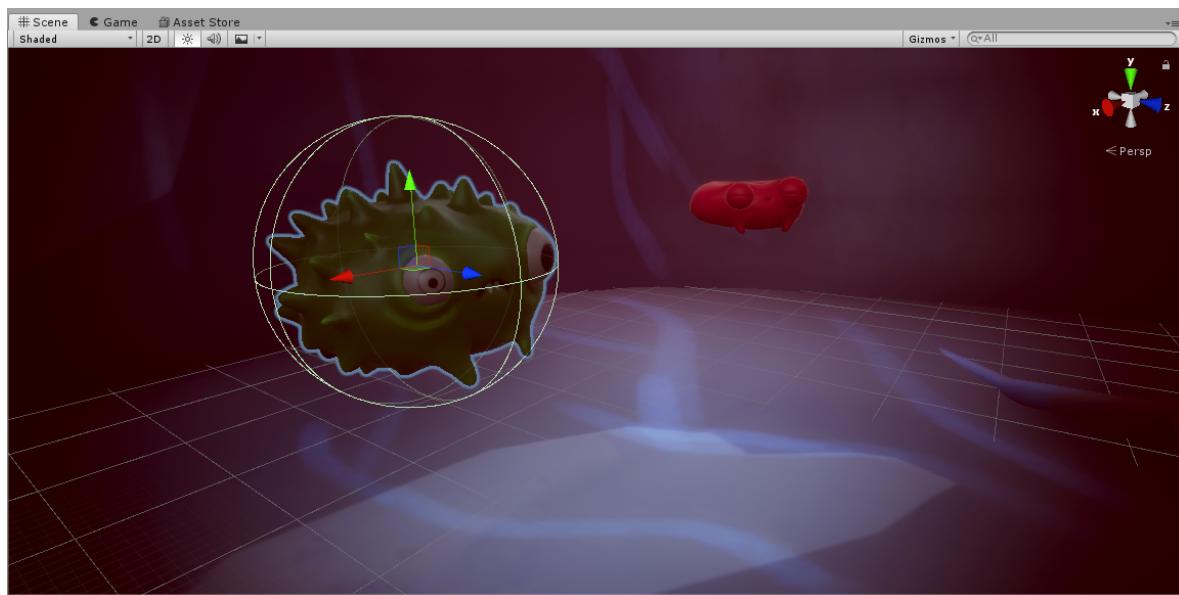
这将使新细菌成为游戏中的物理对象。禁用“**使用重力**”会使细菌漂浮，而启用“**Is Kinematic**”将阻止子弹（或其他任何东西）移动。

6. 再次单击**添加组件**，然后搜索**Sphere Collider**。

添加组件。



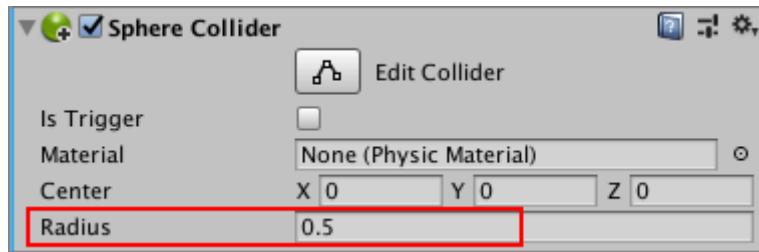
一球撞机将确保胚芽可击;有时称为“hitbox”。在“场景”视图中，对撞机将作为细菌周围的绿色球体。



7. 在**中心**字段中，将Y设置为0.95。这将确保对撞机位于细菌的中心，而不是在细菌的下方。

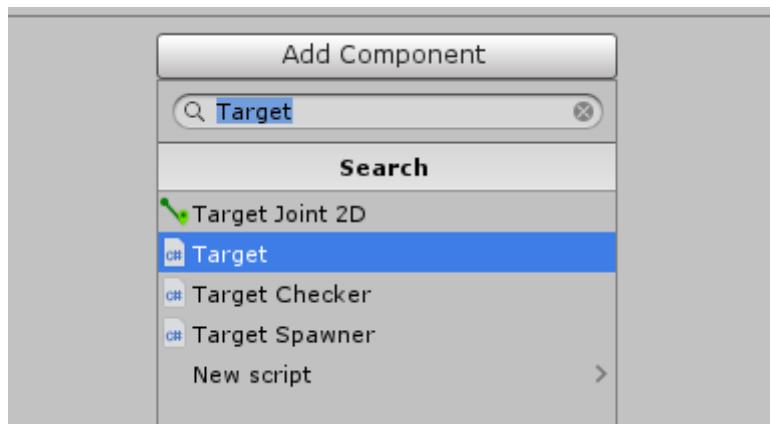
8. 更改对撞机的尺寸：

- 将鼠标悬停在“检查器”窗口中的“半径”一词上。
- 单击并向右或向左拖动鼠标以增加或减小点击框的大小。



增大对撞机将使细菌更容易被击中，减小对撞机将使其更具挑战性。

9. 单击添加组件，然后搜索并添加目标组件。



该组件添加了一个自定义脚本，使脚本成为目标。添加完成后，更改目标的“生命值”和“点值”，然后再次测试游戏。完成后，请确保保存更改。

3.2.5 添加目标动画和声音

为确保在击打细菌时播放正确的音频，需要将GameObjects添加到Target组件：

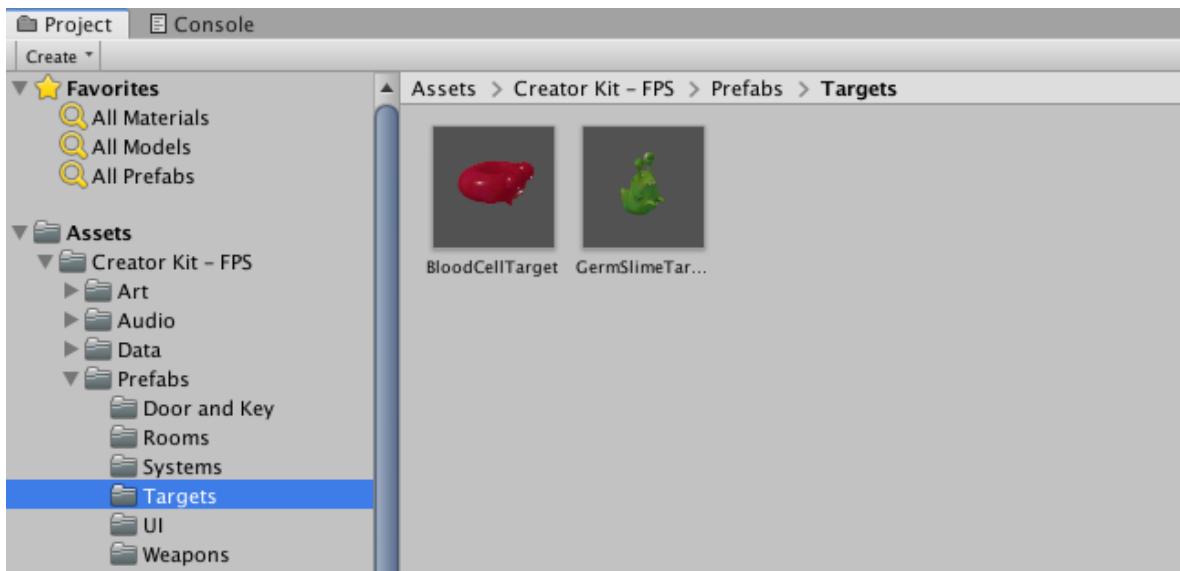
1. 在“层次结构”中，单击GermSpike GameObject 左侧的灰色箭头。这将显示其子GameObjects：HitPlayer 和IdlePlayer 。
2. 将HitPlayer GameObject从“层次结构”拖到检查器“目标”组件中的“**命中播放器**”字段（在“音频”标题下）。
3. 将IdlePlayer GameObject从“层次结构”拖动到“检查器” 中“目标”组件中的“**空闲播放器**”字段（在“音频”标题下）。
4. 保存您的更改。现在您已经添加了正确的声音，有一种简单的方法可以复制GermSpike目标：创建自己的Prefab。

3.2.6 将目标变成预制件

预制件是标准化游戏中不同对象的好方法：链接场景中预制件模板的所有副本。这意味着，如果您更改胚芽预制件的对象值，则场景中该胚芽的每个副本都会更改以匹配它。

要将新的细菌目标转化为预制件：

1. 在“项目”窗口中，转到“**资产/创建者套件**FPS / Prefabs / Targets”。



2. 将GermSpike GameObject 从“层次结构”窗口拖放到“项目”窗口。
3. 将出现一个对话框，询问您是否要保存原始预制件或预制件变体-选择**原始预制件**。

Project文件夹中有一个新的Prefab，您可以使用它将细菌的多个副本添加到场景中。

3.3 关卡设计

3.3.1 设计自己的关卡

现在可以向场景中添加更多目标，这个一室一厅的世界变得非常拥挤是时候扩展游戏的空间了。

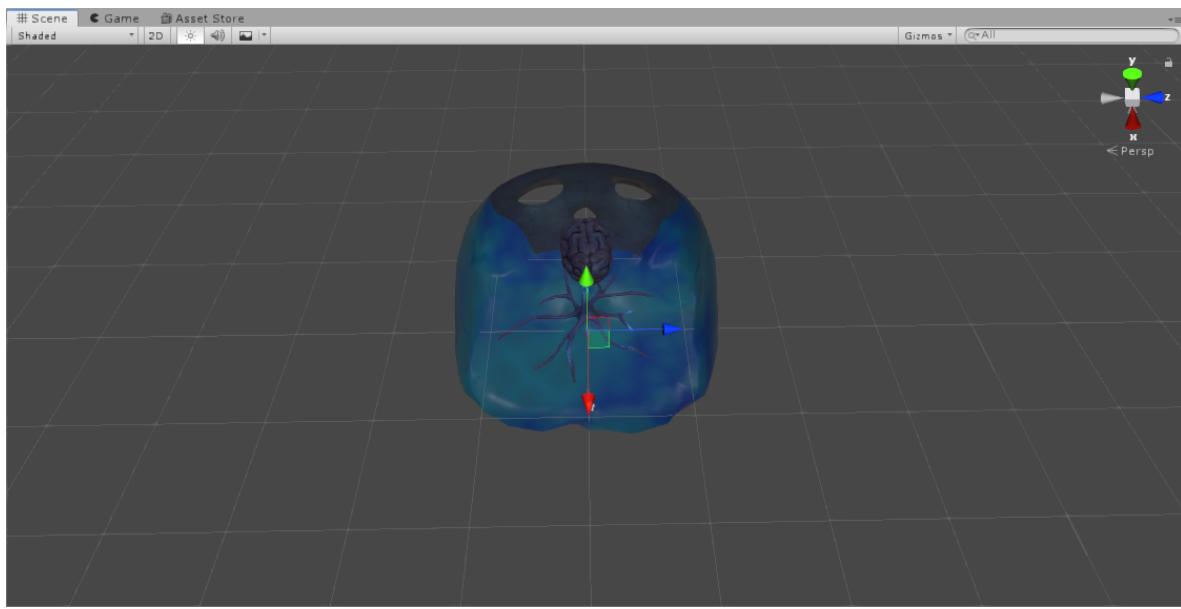
- 为游戏设计一个完整的关卡
- 添加锁定的门和钥匙以打开它们

3.3.2 关闭音频小控件

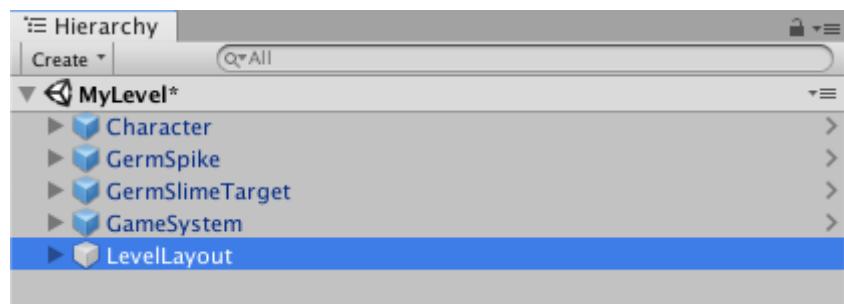
该Creator Kit包含一个简单的关卡编辑器，可以使用该编辑器将通过门连接的房间“捕捉”在一起。为了使此操作简单易行，需要关闭两个在“场景”视图中可见的Gizmos（图形叠加层）。

要关闭Gizmos，请执行以下操作：

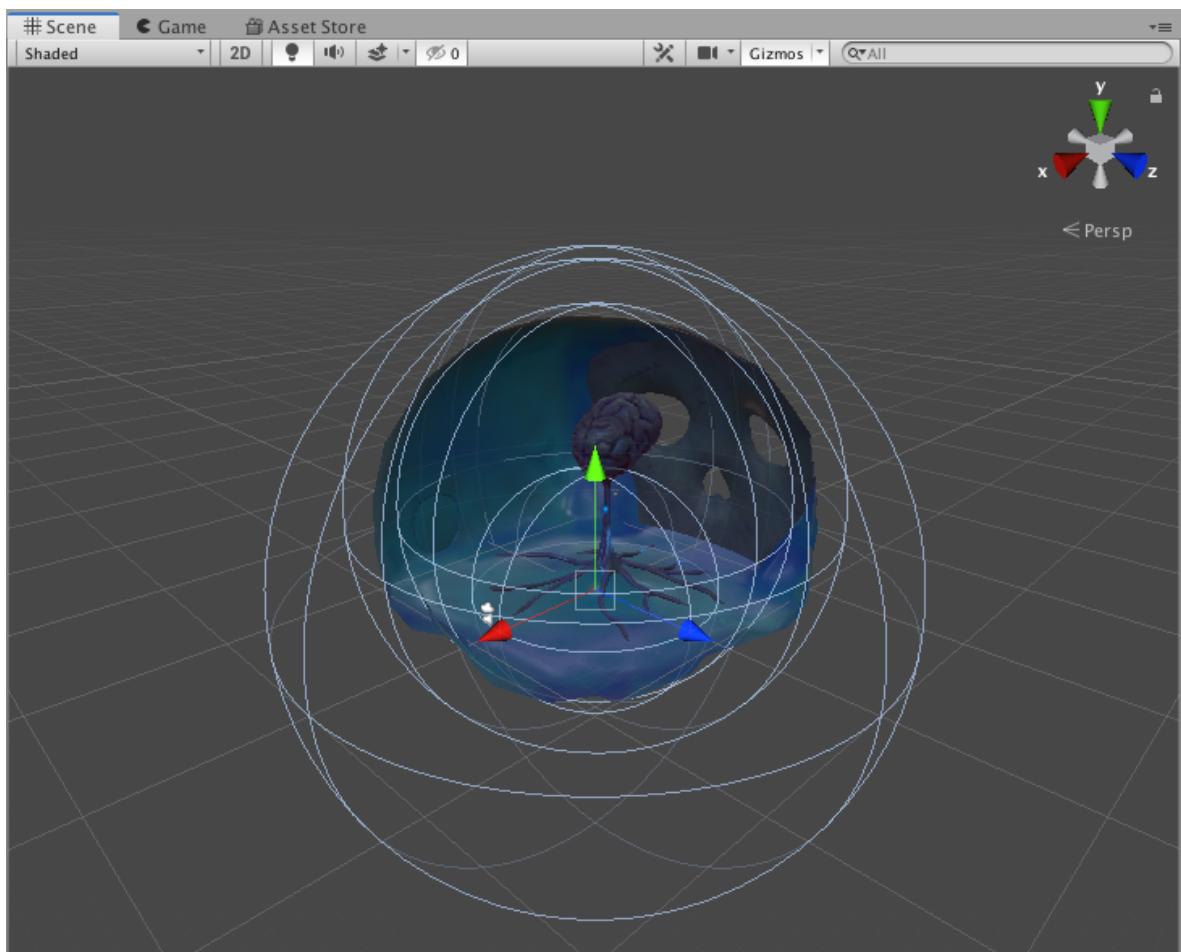
1. 在“场景”视图中，进行缩小，可以看到当前正在工作的房间被默认的灰色空间包围。



2. 在“层次结构”窗口中，选择“LevelLayout GameObject”。

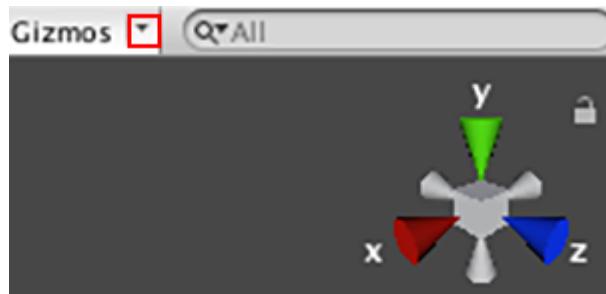


执行此操作时，默认的大脑空间周围将出现蓝色球体：



这些球体代表“音频混响区”和“音频源”，它们是默认情况下附加到所有房间的组件。球体会妨碍编辑关卡布局，因此您将禁用它们。

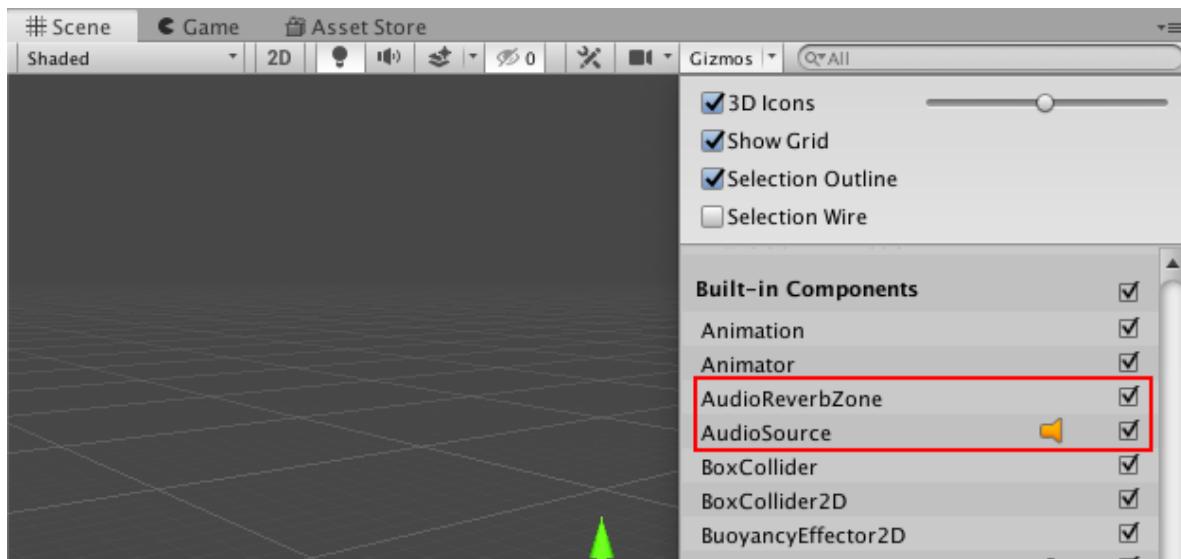
3. 在“场景”视图的右上角，单击Gizmos按钮旁边的向下箭头。



注意：单击Gizmos按钮本身将禁用所有Gizmo图标，这不是您需要在此Creator Kit中执行的操作。

4. 在菜单中，向下滚动列表并禁用以下两个内置组件复选框：

- AudioReverbZone
- 音源



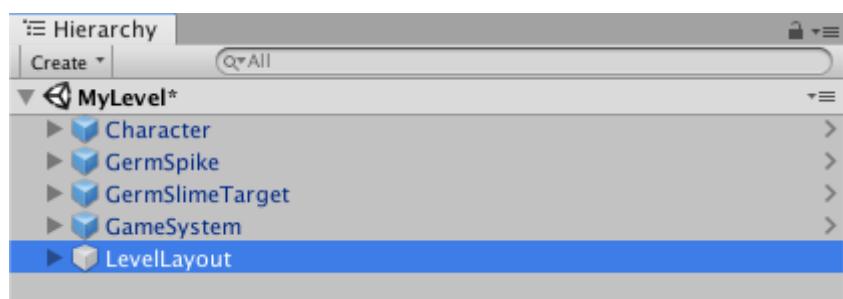
就是这样-现在，在“场景”视图中为关卡放置房间和走廊将变得更加容易。

提示：关闭所有音频混响区域和音频源的Gizmos。需要随时查看其范围，可以通过再次启用其复选框来打开Gizmos。

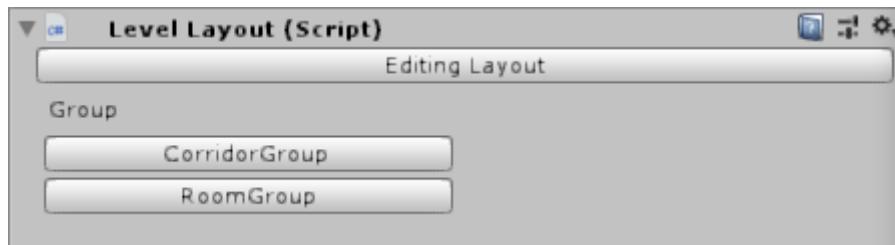
3.3.3 添加新房间和走廊

关闭了Gizmos，则可以使用关卡布局编辑器来设计自己的游戏关卡：

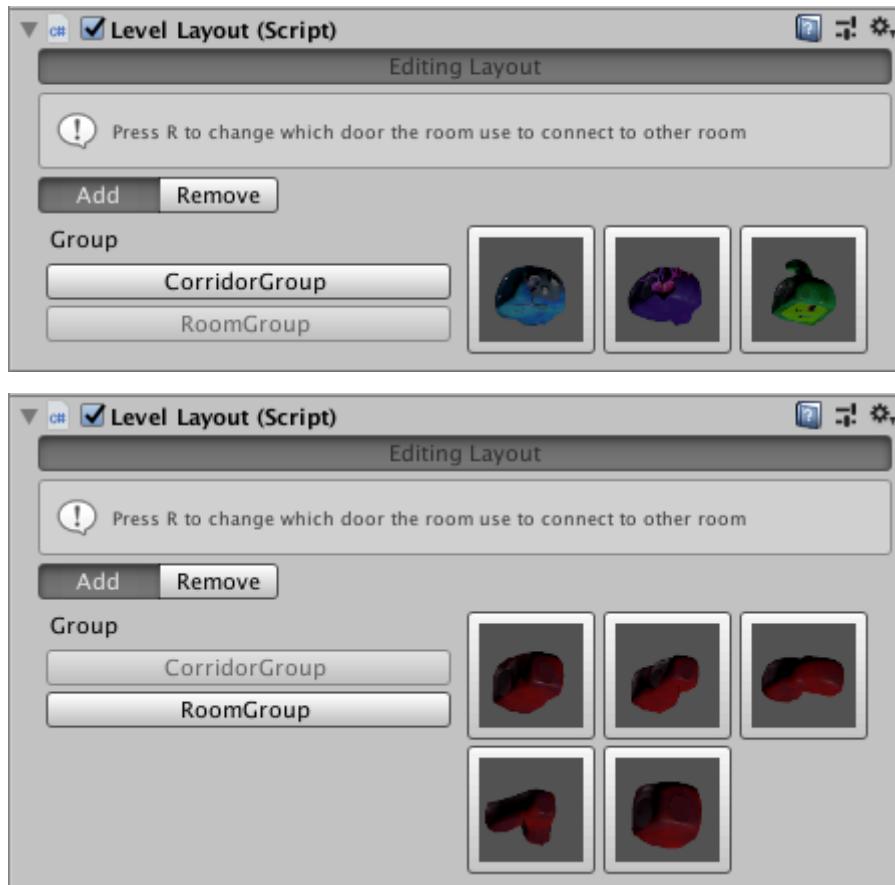
1. 在层次结构中，检查是否仍选择了LevelLayout GameObject。



2. 在“检查器”窗口中，找到“**关卡布局**”组件。点击添加。然后单击“**走廊组**”以添加走廊，或单击“**室组**”以添加“房间”。



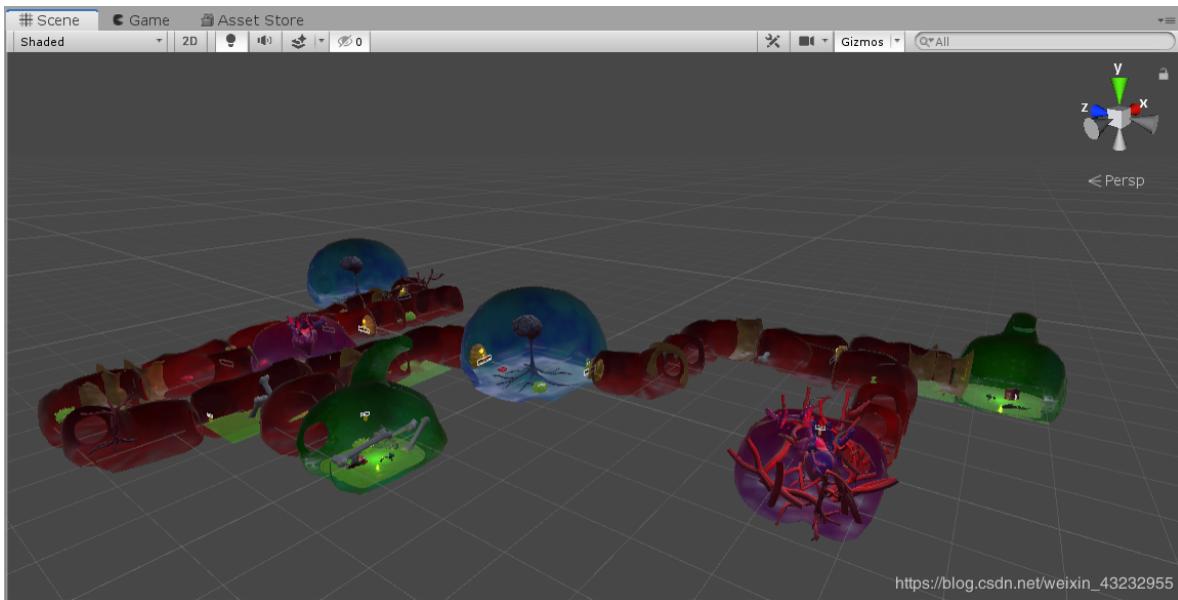
这将显示可选走廊或房间的缩略图：



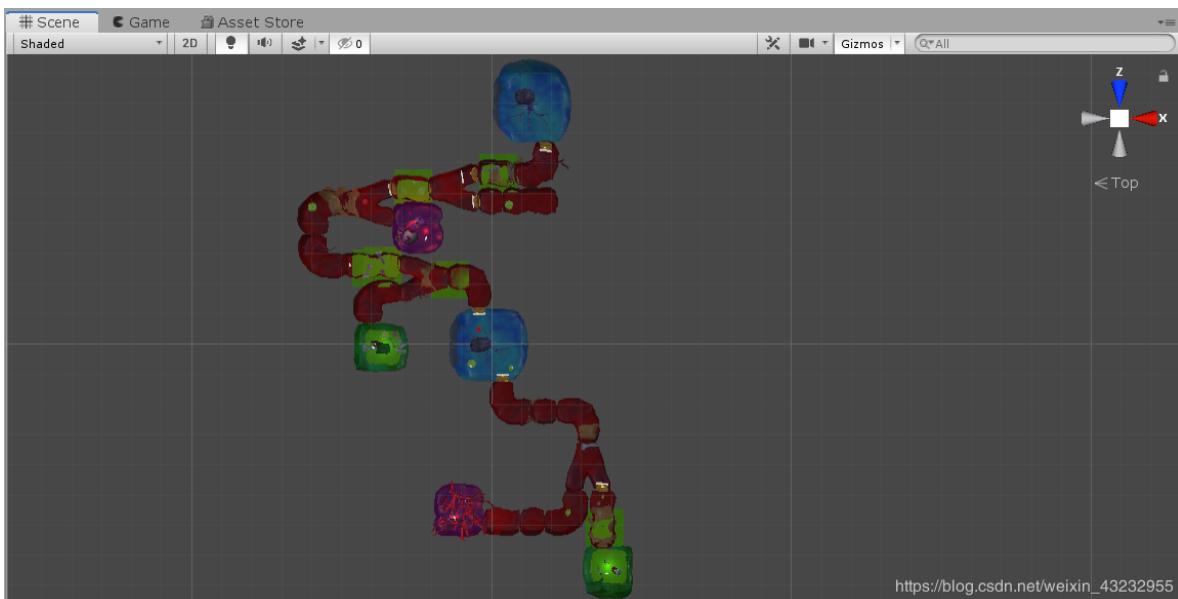
3. 单击一个可选房间，将：

- 将鼠标移到“场景”视图上时，将出现在场景中
- 在编辑器中自动捕捉到网格系统
- 按键盘上的R来旋转房间并更改它要锁定到的门

地图预览



https://blog.csdn.net/weixin_43232955



https://blog.csdn.net/weixin_43232955

4. 如果输入错误或要删除房间，请单击组件中的“**删除**”按钮。完成此操作后，将光标移至“场景”视图中的房间和走廊上方将突出显示它们。单击一个房间将其删除。

5. 完成关卡设计的**编辑后**，再次单击“检查器”窗口中的“**编辑布局**”按钮以禁用关卡编辑器工具。

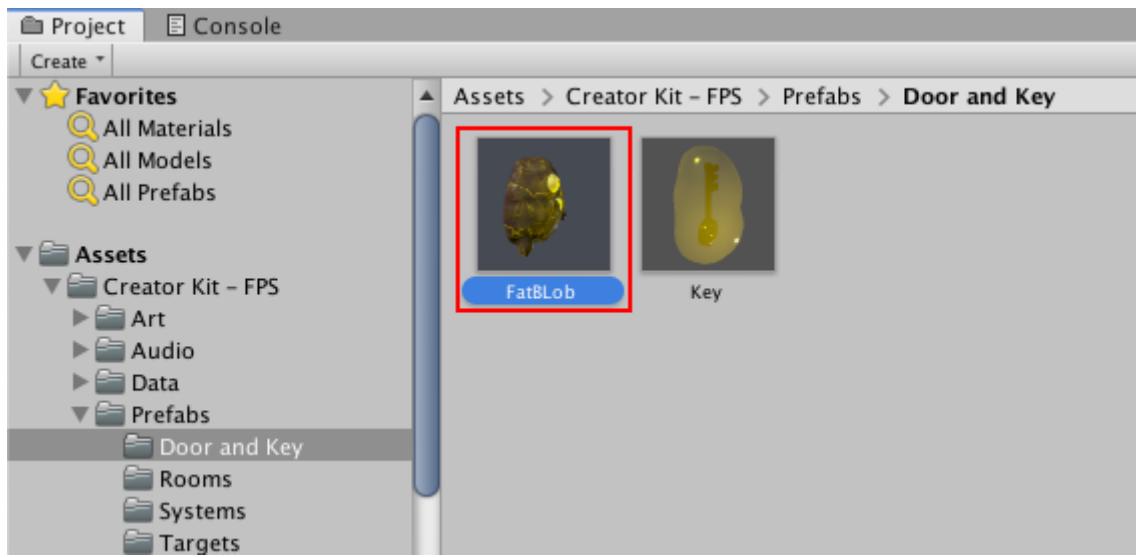
6. 保存更改，然后再次玩游戏以测试新关卡设计。

3.3.4 添加上锁的门

此刻，医生可以在身体上随处奔跑。为了使他们的任务更具挑战性，您可以锁定房间的门，直到玩家找到特定的“钥匙”为止，来继续通过场景来消灭细菌

首先，向房间添加一扇锁着的门：

1. 在Project窗口中，转到*Assets / Creator Kit-FPS / Prefabs / Door and Key*。
2. 选择*FatBlob* 预制件。这将充当游戏门的锁。

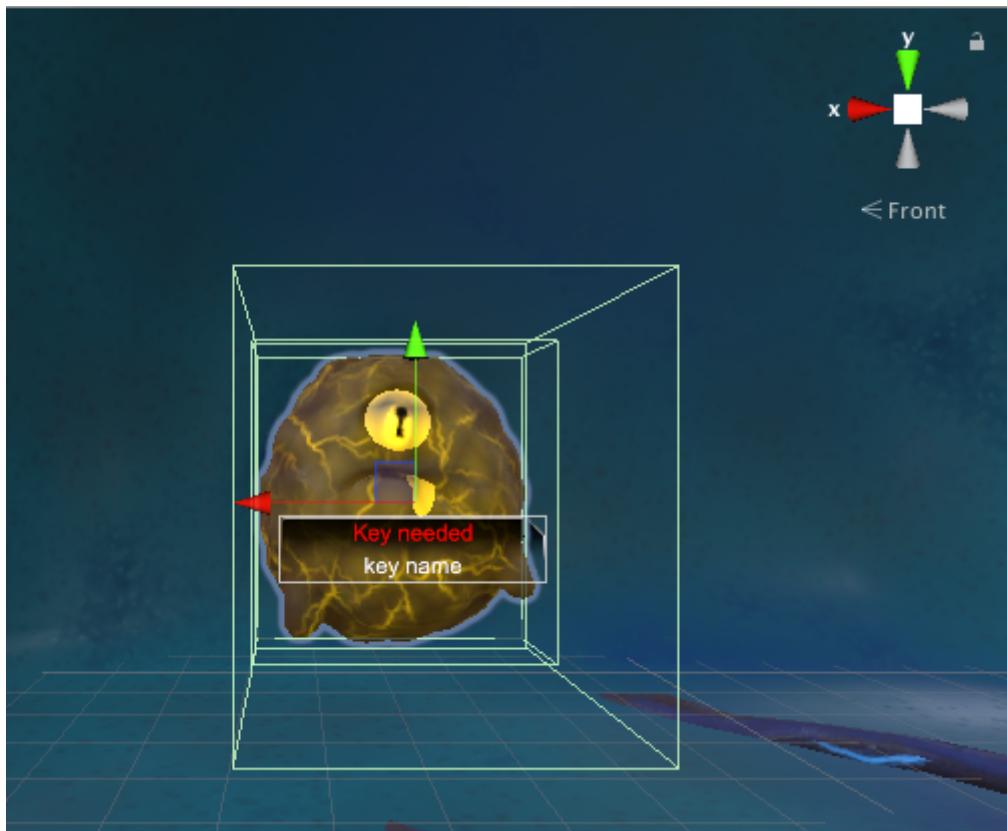


3. 将FatBlog预制件从“项目”窗口拖动到“场景”视图中，将其放置为挡住门口。



锁的正面标有钥匙名称。

锁尚未分配钥匙：

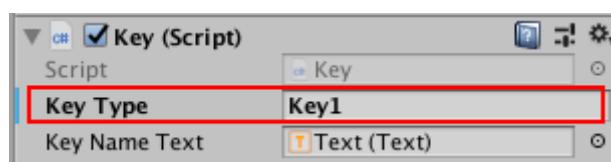


- 保存对游戏的更改。

3.3.5 将钥匙与锁关联

现在，玩家将需要解锁门才能进入关卡。要将钥匙与锁关联起来：

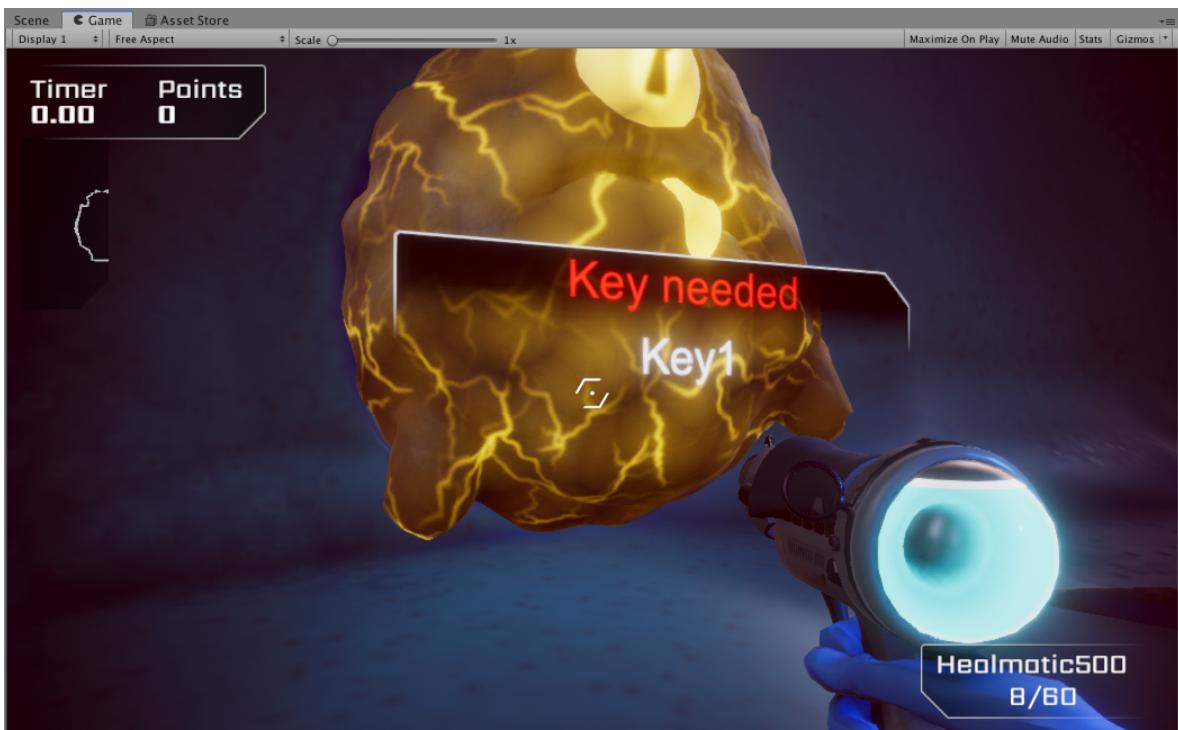
- 在“项目”窗口中，选择“ Key Prefab”。
- 将Key 拖动到FatBlob门锁所在房间的“场景”视图中。
- 在检查器中，找到“**关键**”组件。在“**密钥类型**”字段中，为您的密钥指定一个唯一的名称。



- 在层次结构中，选择FatBlob GameObject。
- 在检查器中，找到其“**锁定**”组件。使用下拉菜单将“**密钥类型**”值设置为您选择的密钥名称。



- 保存游戏，然后在“播放模式”下进行测试。当医生离锁足够近时，将显示其关联钥匙的名称：



领取钥匙后，他们就可以穿过门了。

现在，医生可以在关卡中定位目标和上锁的门，从而使游戏更具挑战性

3.4 武器修改

3.4.1 创建新武器

现在，医生必须面对更具挑战性的目标，所以要增强他们的细菌破坏工具了。

在本部分中，将修改游戏中可用的武器，以便为玩家提供更多选择。

要创建新的武器预制件：

1. 在“项目”窗口中，转到“ 资产/创建者套件-FPS / Prefabs / 武器”
2. 选择GermOBlaster 预制件。通过按Ctrl + D 进行复制以进行复制



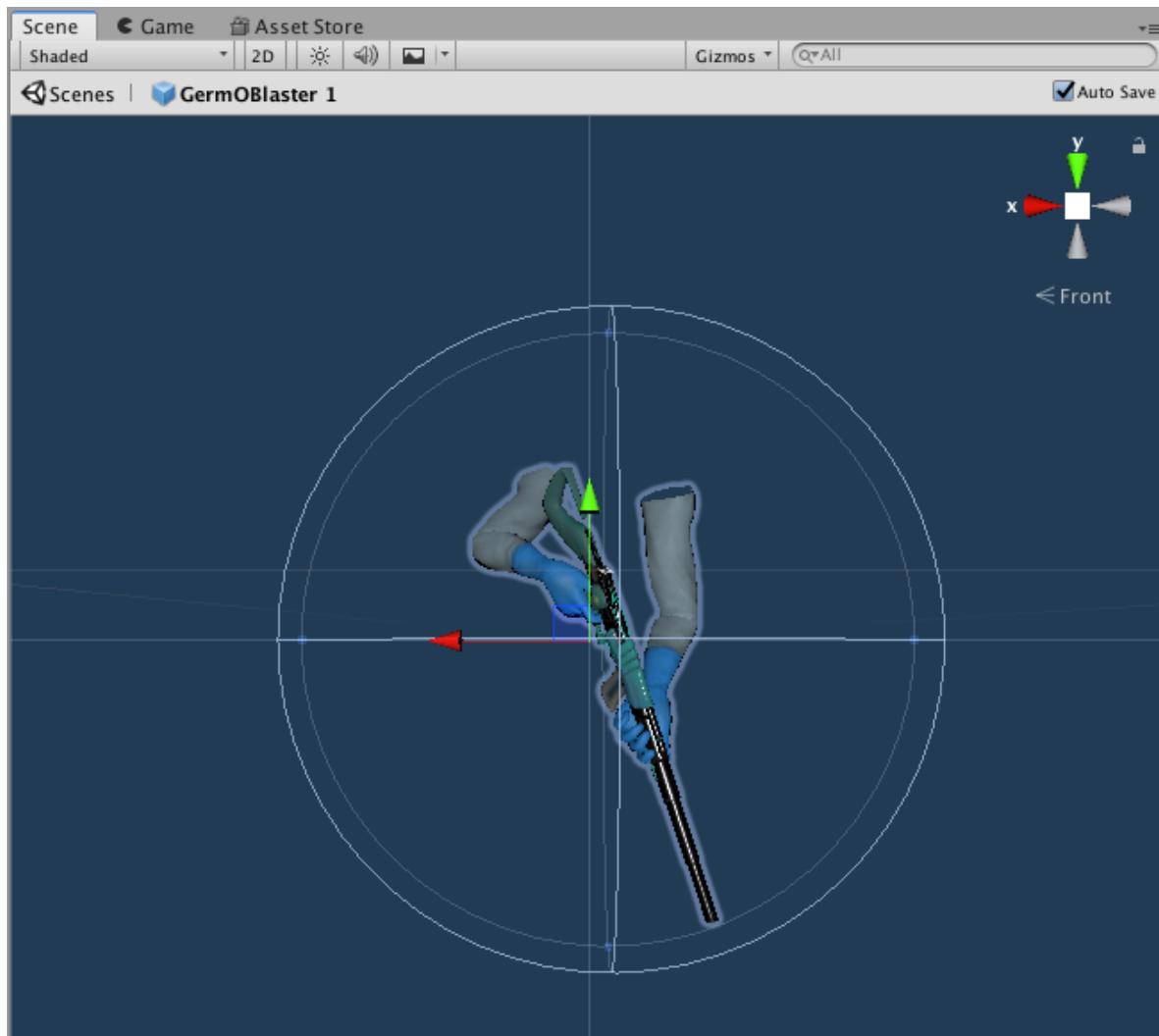
3. 单击副本名称，或选择它，然后按F2 。给新武器起一个不同的名字。

4. 保存您的更改。

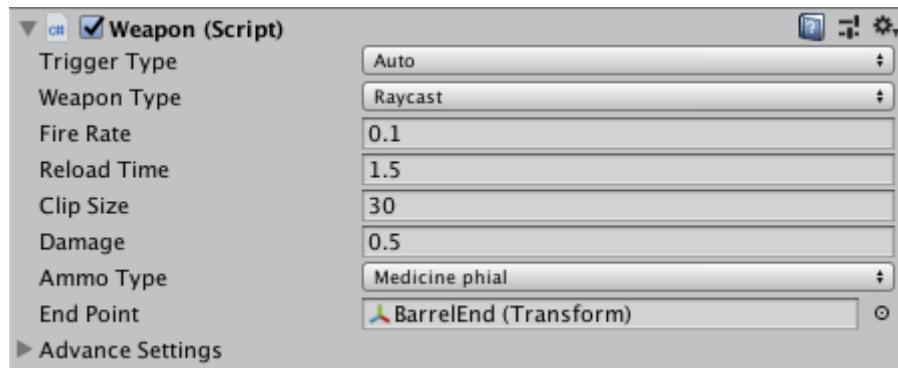
3.4.2 调整新武器预制件

原始的GermOBlaster具有自动高速发射功能，带有大量药弹。让我们每次触发按一次就可以使新武器弹药更少，但伤害更高：

1. 双击新武器，或在选择副本后单击“检查器”窗口中的“打开预制”按钮。Unity Editor现在处于“预制模式”，并将在“场景”视图中显示武器



2. 在检查器中，找到“**武器**”组件。这是为此Creator Kit编写的自定义脚本，具有许多不同的可配置值



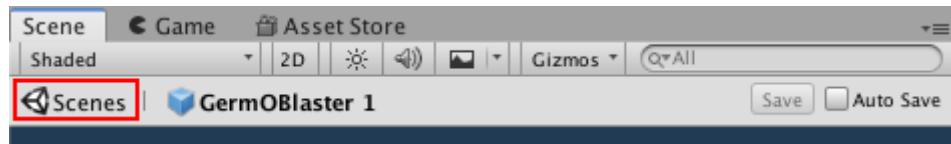
3. 使用下拉菜单将**触发器类型**更改为**手动**。现在，仅在按下扳机时，武器才会发射一次，而不是尽可能快地发射。

4. 将**剪辑大小**值从30 更改为8，以减少单个重新加载剪辑中的弹药量

5. 将**伤害**值从0.5 更改为5。这意味着，与GermOBlaster相比，使用此武器的每次命中对细菌的伤害更大

6. 现在，您已经创建了一种新型武器，请使用键盘快捷键保存“预制”，或单击“场景”视图右上角的“保存”按钮

7. 通过单击“**场景**”视图左上角的“**场景**”退出“预制模式”



3.4.3 使玩家可以使用武器

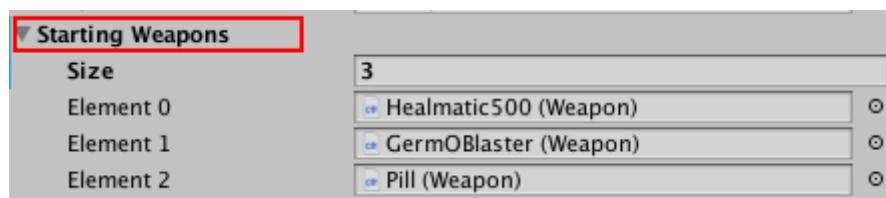
现在已经成功创建了另一种消灭细菌的武器，将其提供给玩家：

1. 在层次结构中，选择**角色** GameObject。

2. 在“检查器”窗口中，找到**控制器**组件

这是另一个自定义脚本，可处理所有玩家输入

3. 选择名称左侧的小箭头，展开“**开始武器**”条目



4. 将**大小**从3 更改为4。这将在级别开始时为玩家提供四个武器

5. 将新武器Prefab从“项目”窗口拖动到“检查器”中已出现的“**元素3**”字段。

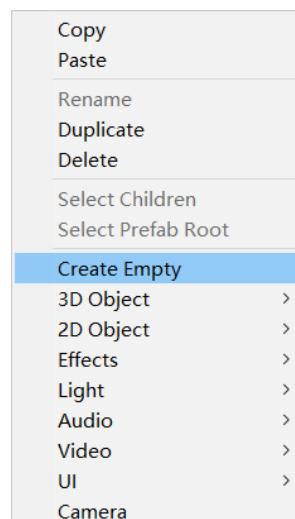
6. 保存更改

3.5 游戏终点及整体逻辑

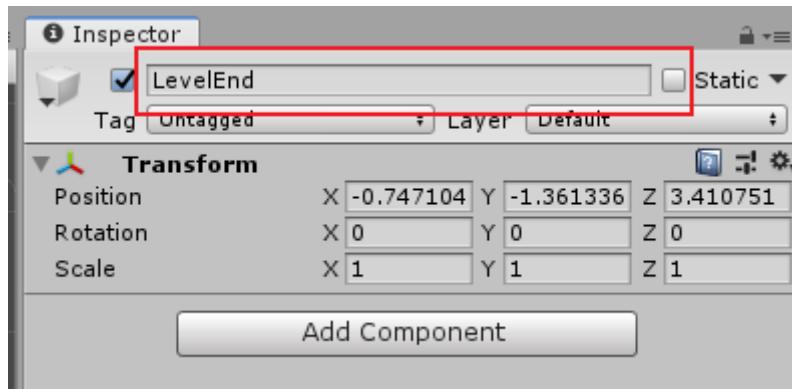
3.5.1 创建一个LevelEnd

首先，创建一个LevelEnd GameObject：

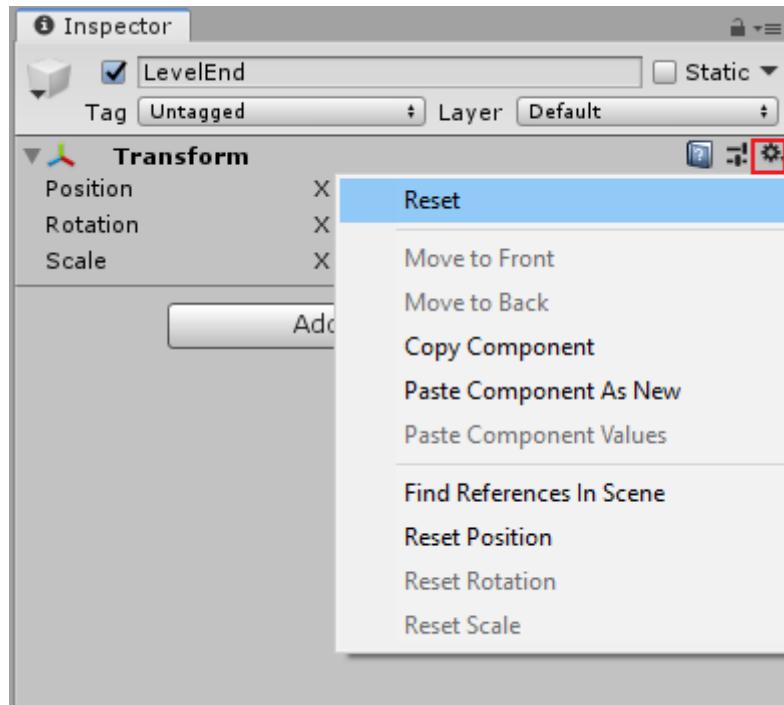
1. 在“层次结构”窗口中的空白处单击鼠标右键，然后选择“**创建空白**”。



2. 选中GameObject，然后转到“**检查器**”窗口。使用标题中的文本字段将其重命名为“**LevelEnd**”。

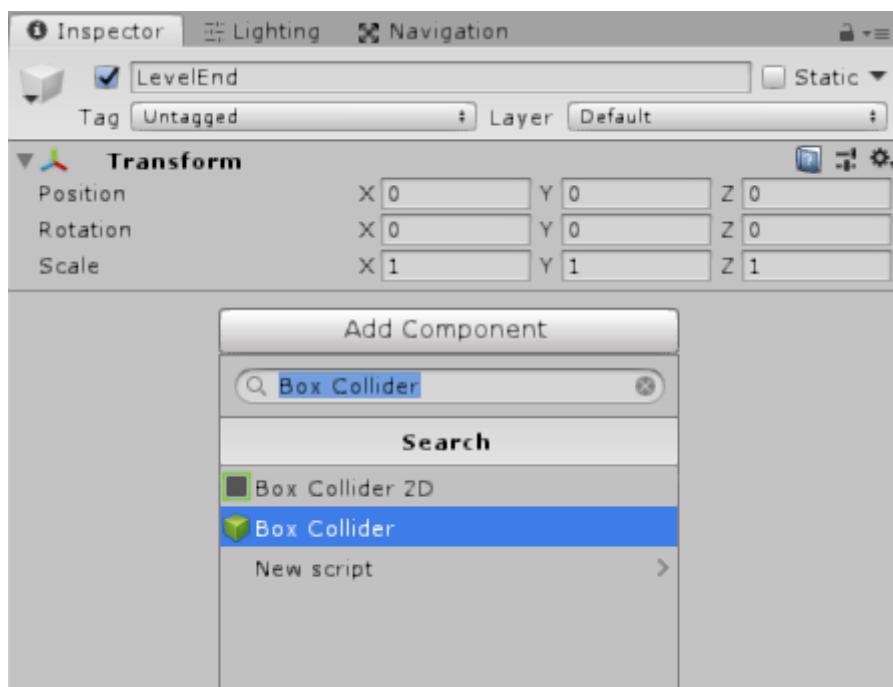


3. 接下来，您需要确保LevelEnd在关卡的中间。单击“**变换**”组件左上方的小齿轮图标。选择重置。



4. 将光标移到“场景”视图上，然后按键盘上的F以将焦点对准LevelEnd GameObject。

5. 在检查器中，单击“**添加组件**”按钮。搜索“**Box Collider**”，然后将此组件添加到LevelEnd。



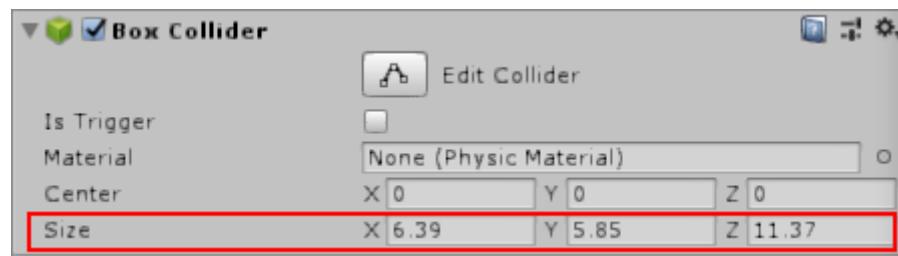
6. 一个绿色的线框框将出现在“场景”视图中，表明LevelEnd具有对撞机。



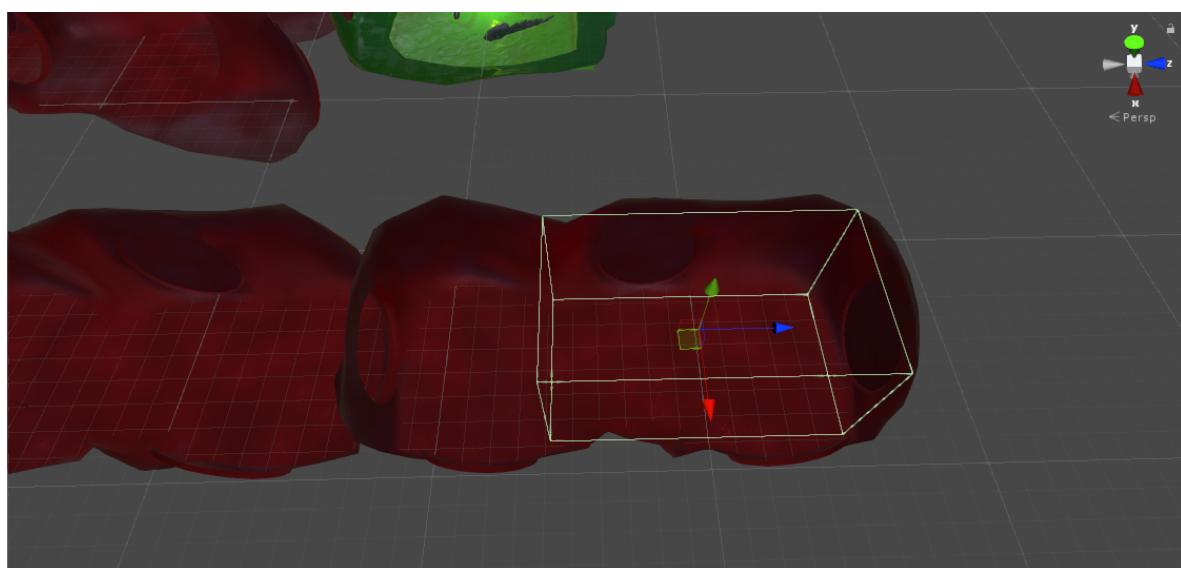
使用**变换工具**将对象放置在您希望关卡结束的位置。

房间或走廊的尽头是执行此操作的好地方。

7. 可以使用Box Collider组件中的**Size**参数来调整Collider的**大小**。一次将鼠标悬停在一个轴上，然后单击并向右或向左拖动以增加或减小Collider的大小。



LevelEnd GameObject已移动到房间的尽头。对撞机已调整大小，以填充大部分可用空间。



8. 保存您的更改。

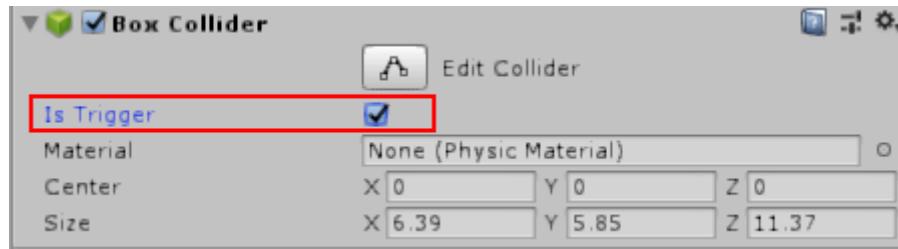
3.5.2 添加触发器

现在，需要添加一个游戏结束的触发器。

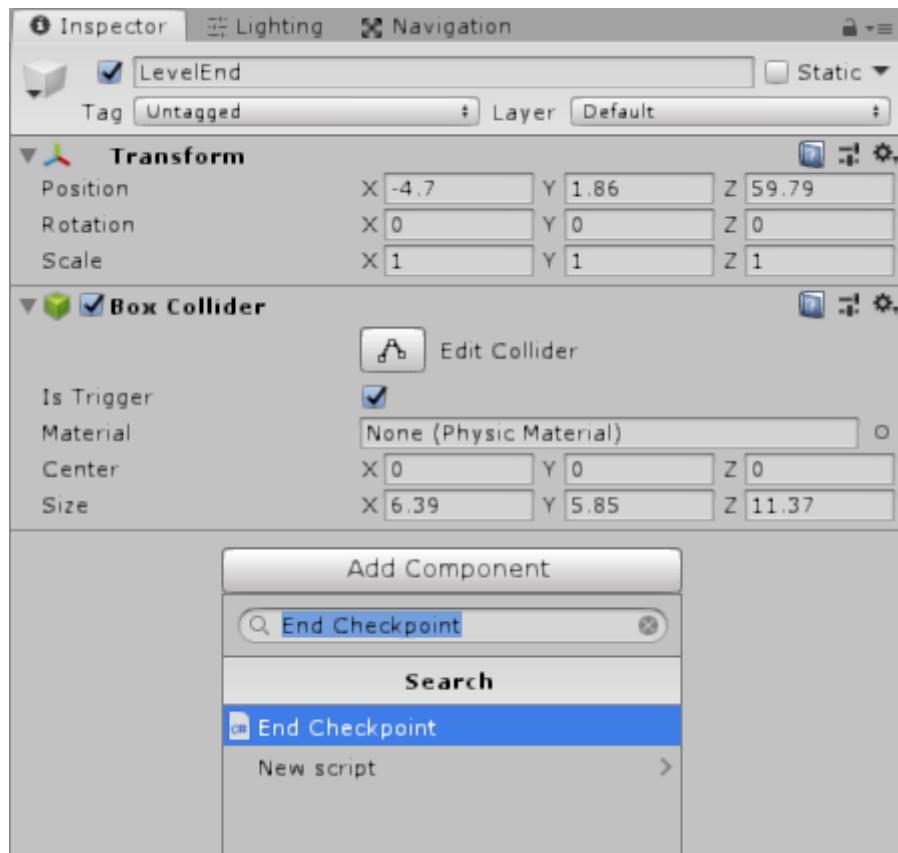
触发器是游戏的基本构建块，可用于执行许多不同的操作。例如，当玩家到达某个点，播放声音或加载新游戏级别时，触发器可以启动过场动画。关卡设计的重要部分是放置碰撞器，该碰撞器将在玩家到达特定位置时触发游戏事件。

为了使玩家能够完成关卡，需要将LevelEnd Box Collider设置为游戏最终屏幕的触发器：

1. 在检查器中，转到Box Collider组件。启用**Is Trigger**复选框。



2. 单击**添加组件**按钮。搜索“End Checkpoint”，然后将此组件添加到LevelEnd。



3. 保存更改。现在，按“播放”并通过移动到放置在关卡中的Box Collider中来测试触发器。当进入该空间时，游戏将结束并显示结果屏幕

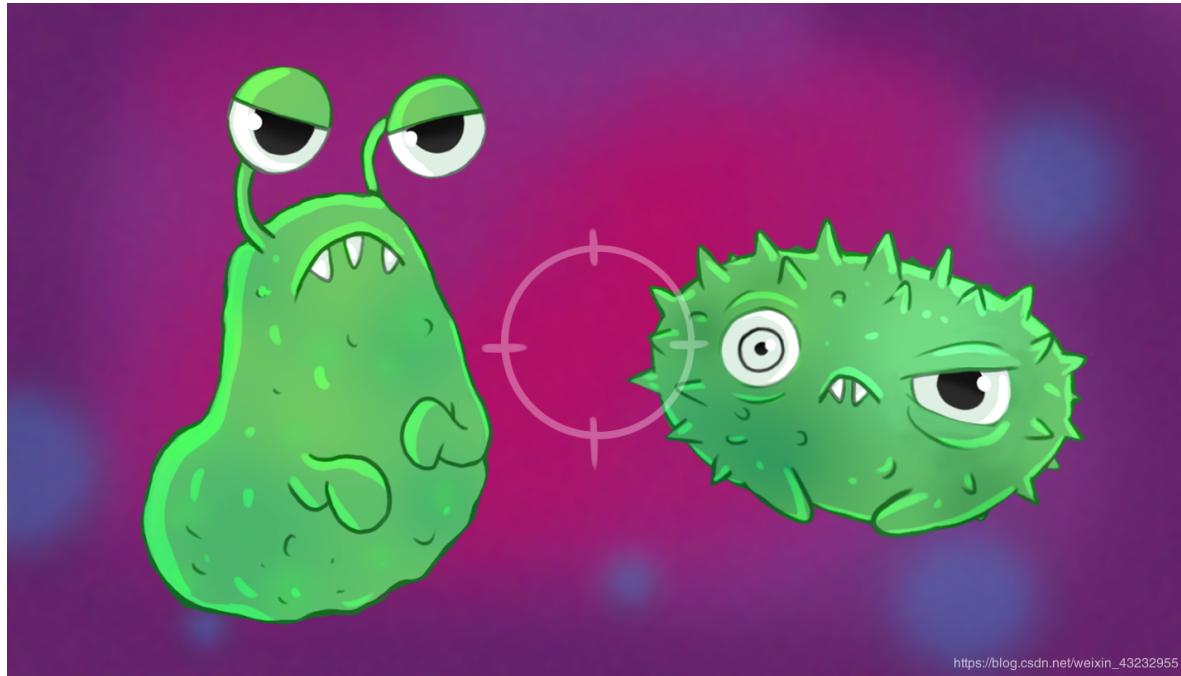
3.6 开始UI界面及发布游戏

3.6.1 开始UI界面

现在，游戏场景已经搭建完成，开始，结束，细菌放置及设置，玩家移动，开启关卡，钥匙藏放，开门逻辑，模型碰撞检测 等等步骤全部测试完毕。

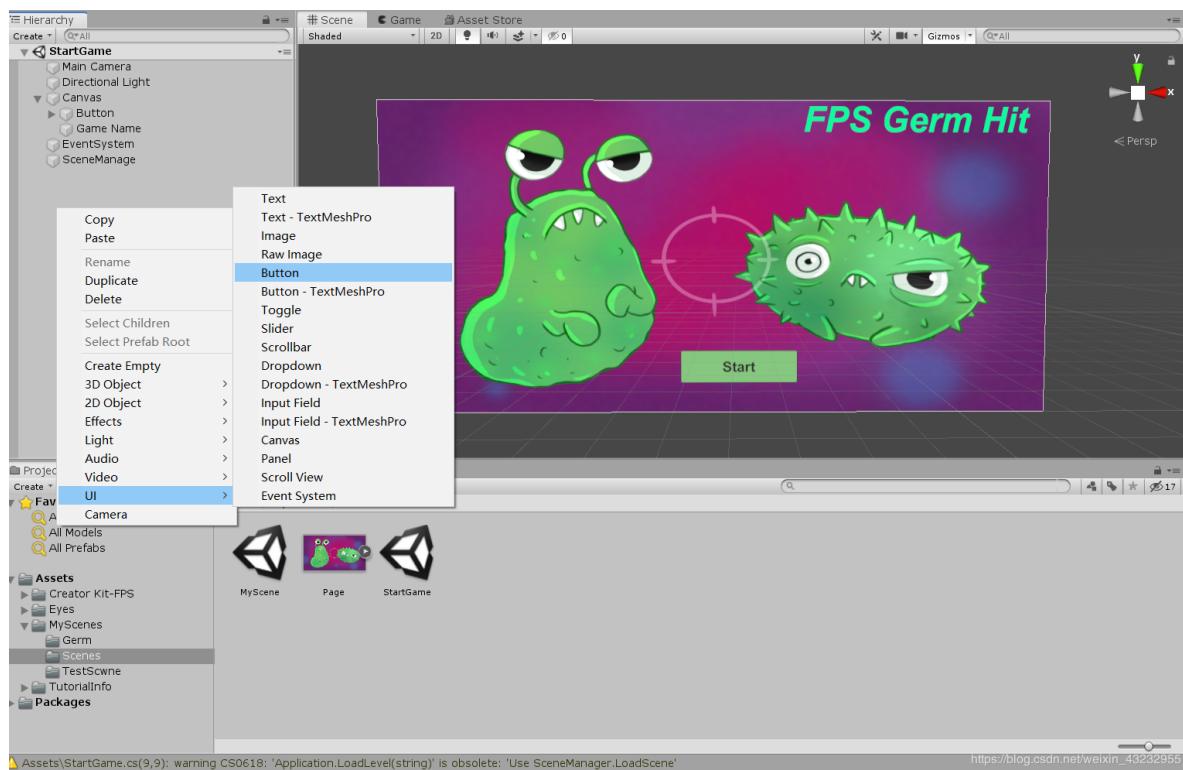
至此共计测试32次，感谢每一位小组成员的合作和付出

- 现在，要将这个封面作为游戏开始的UI界面



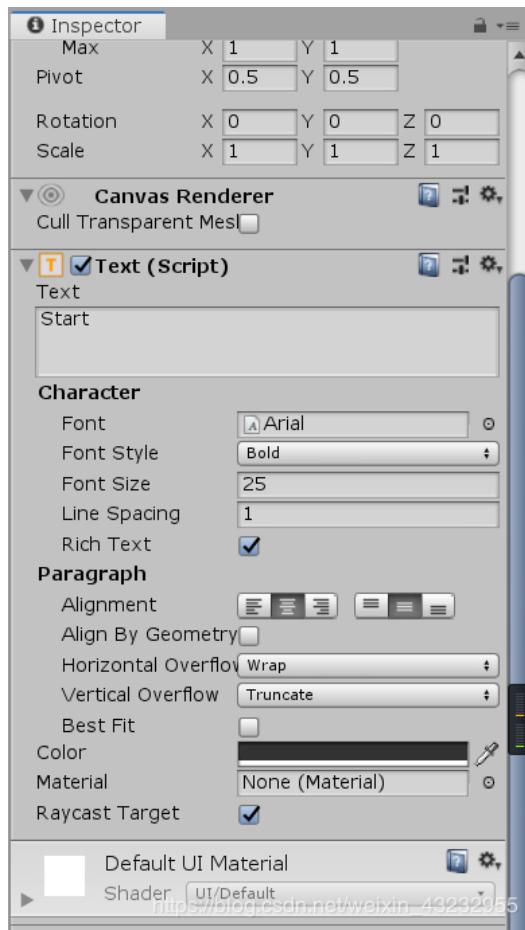
https://blog.csdn.net/weixin_43232955

I. 创建 Button

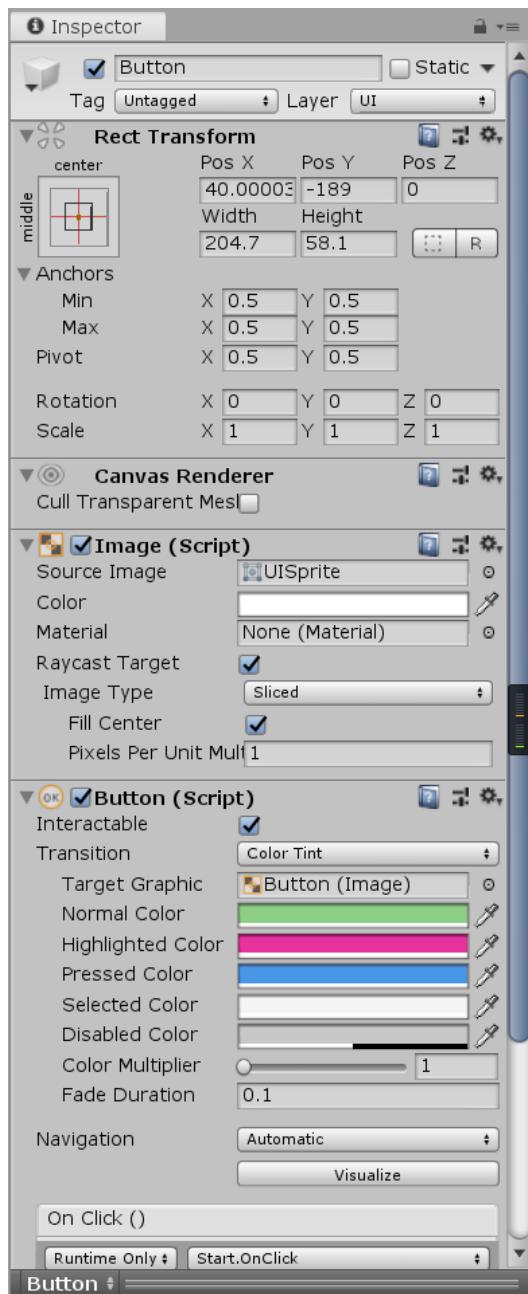


https://blog.csdn.net/weixin_43232955

点击Button 下的 Text，在右侧 Inspector 面板中，添加 Start 按钮内容，修改文字大小



选中Button，修改Button颜色，点击时颜色及选择时颜色

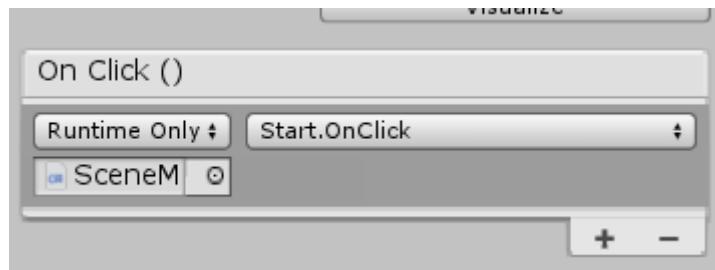


并且为游戏添加名字，我们的游戏名为FPS Germ Hit



II. 点击时场景切换

从UI界面切换到游戏中，在Button上挂载点击事件，此代码为 **SceneManage**



- 调整图片大小，按钮大小，文本文字大小
- 测试点击之后场景是否跳转

再次测试，游戏整体完成

第四章 碰撞检测

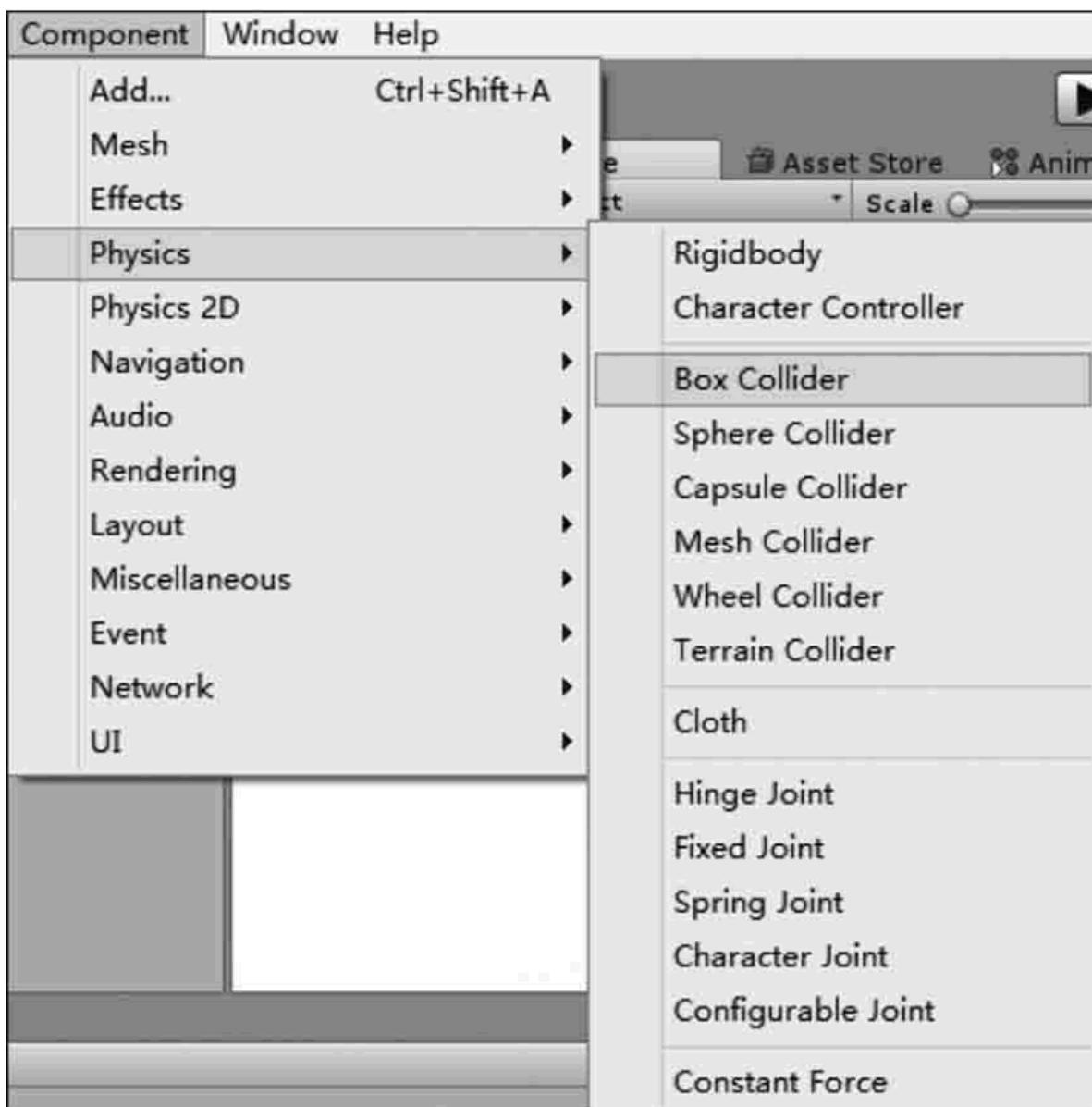
4.1 碰撞检测

在游戏制作过程中，游戏对象要根据游戏的需要进行物理属性的交互。因此，的物理组件为游戏开发者提供了碰撞体组件。碰撞体是物理组件的一类，它与刚体一起促使碰撞发生。

一个立方体会得到一个 Box Collider（立方体碰撞体），一个球体会得到一个 Sphere Collider（球体碰撞体），一个胶囊体会得到一个 Capsule Collider（胶囊体碰撞体）等。

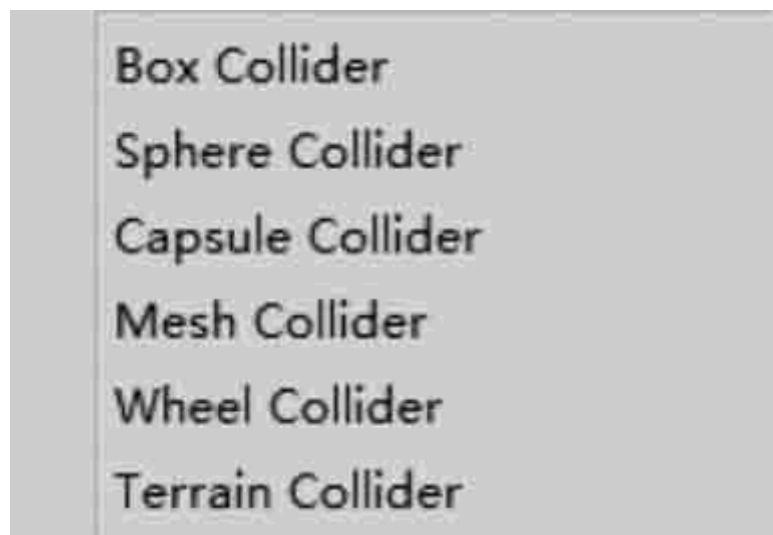
添加碰撞体

在 Unity 3D 的物理组件使用过程中，碰撞体需要与刚体一起添加到游戏对象上才能触发碰撞。值得注意的是，刚体一定要绑定在被碰撞的对象上才能产生碰撞效果，而碰撞体则不一定要绑定刚体。



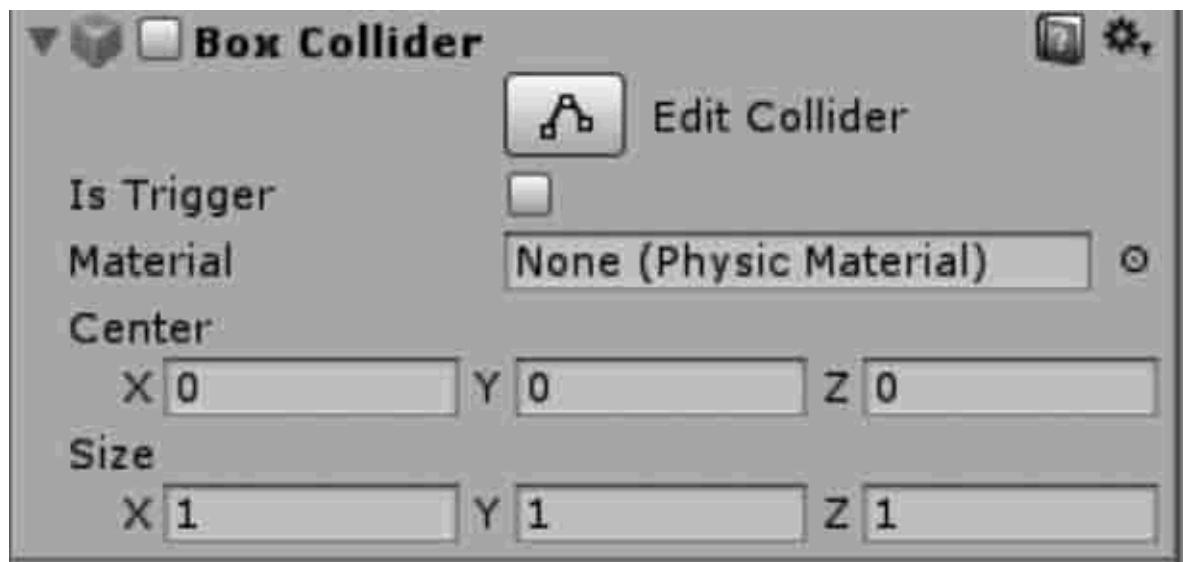
碰撞体选项设置

Unity 3D 为游戏开发者提供了多种类型的碰撞体资源，如下图所示。当游戏对象中的 Rigidbody 碰撞体组件被添加后，其属性面板中会显示相应的属性设置选项，每种碰撞体的资源类型稍有不同，具体如下。



1) Box Collider

Box Collider 是最基本的碰撞体，Box Collider 是一个立方体外形的基本碰撞体。一般游戏对象往往具有 Box Collider 属性，如墙壁、门、墙以及平台等，也可以用于布娃娃的角色躯干或者汽车等交通工具的外壳，当然最适合用在盒子或是箱子上。



参数	含义	功能
Is Trigger	触发器	勾选该项，则该碰撞体可用于触发事件，并将被物理引擎所忽略
Material	材质	为碰撞体设置不同类型的材质
Center	中心	碰撞体在对象局部坐标中的位置
Size	大小	碰撞体在X、Y、Z方向上的大小

如果 Is Trigger 选项被勾选，该对象一旦发生碰撞动作，则会产生 3 个碰撞信息并发送给脚本参数，分别是 OnTriggerEnter、OnTriggerExit、OnTriggerStay。

Sphere Collider 是球体形状的碰撞体，如下图所示。

当游戏对象的物理形状是球体时，则使用球体碰撞体，如落石、乒乓球等游戏对象。

2) Capsule Collider

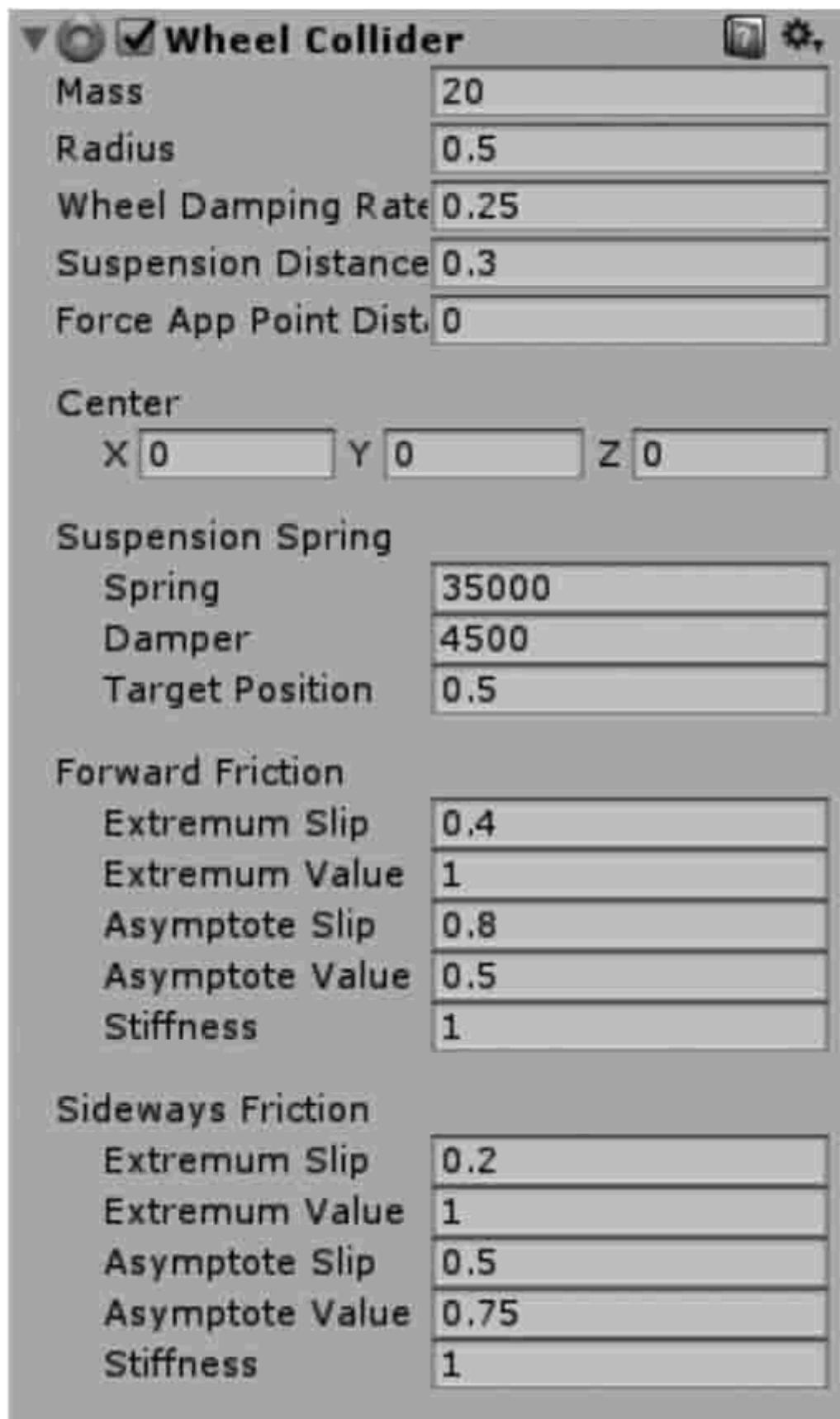
Capsule Collider 由一个圆柱体盒两个半球组合而成，Capsule Collider 的半径和高度都可以单独调节，可用在角色控制器或其他不规则形状的碰撞结合来使用。通常添加至 Character 或 NPC 等对象的碰撞属性，如下图所示，具体参数如下表所示。

3) Mesh Collider

Mesh Collider（网格碰撞体）根据 Mesh 形状产生碰撞体，比起 Box Collider、Sphere Collider 和 Capsule Collider，Mesh Collider 更加精确，但会占用更多的系统资源。专门用于复杂网格所生成的模型，如下图所示，具体参数如下表所示。

4) Wheel Collider

Wheel Collider (车轮碰撞体) 是一种针对地面车辆的特殊碰撞体，自带碰撞侦测、轮胎物理现象和轮胎模型，专门用于处理轮胎，如下图所示，具体参数如下表所示。



参数	含义	功能
Mass	质量	用于设置 Wheel Collider 的质量
Radius	半径	用于设置碰撞体的半径大小
Wheel Damping Rate	车轮减震率	用于设置碰撞体的减震率
Suspension Distance	悬挂距离	该项用于设置碰撞体悬挂的最大伸长距离，按照局部坐标来计算，悬挂总是通过其局部坐标的 Y 轴延伸向下
Center	中心	用于设置碰撞体在对象局部坐标的中心
Suspension	悬挂弹簧	用于设置碰撞体通过添加弹簧和阻尼外力使得悬挂达到目标位置
Forward Friction	向前摩擦力	当轮胎向前滚动时的摩擦力属性
Sideways Friction	侧向摩擦力	当轮胎侧向滚动时的摩擦力属性

4.2 通关机制设计

玩家需要在不同的场景中寻找开启钥匙的门，消灭细菌，保卫人体健康，抵达关卡终点。开启新的健康之旅！

第五章 问题及心得

5.1 问题及解决方案

5.1.1 问题

- 素材的收集以及无法导入
- 素材导入之后无法在场景中加载
- 物理组件各种碰撞器的使用
- 碰撞检测以及穿模问题
- 子弹穿模，穿过细菌（无法碰撞检测）
- 地图场景的搭建
- 门和钥匙 Key - Value 的对应
- 开始 Start 场景的无法跳转到 Game 场景

5.1.2 解决方案

I. 基于unity3D场景切换所遇到问题及解决方法

1. 开始 Start 场景的无法跳转到 Game 场景

代码书写和应用错误

以前只是负责用，却不知道里面的具体含义，特定去查看了一下资料，记录下来。

`Application.LoadLevel()` : 同步载入

`Application.LoadLevelAsync()` : 异步载入

`Application.LoadLevelAddictive()` : 同步附加式载入

`Application.LoadLevelAddictiveAsync()` : 异步附加式载入

四种方法进行简要的介绍和分析

同步载入：如果当前场景为A，我们要切换到场景B，unity会在切换场景的时候将场景B中的全部内容（场景B中全部的静态物体，不包含允许代码中Instantiate实例化的物体）都载入到内存中。然后销毁A，显示B因为载入B是同步进行的（也就是在一个线程中），所以当B场景较大的时候执行此方法会感觉到卡顿。

异步载入：这个与第一种情况基本一样，唯一的差别是载入B的过程是异步的（也就是载入新场景的行为是在一个后台线程中进行的。不影响主线程的执行），这样在载入B的时候当前场景A不受影响，能够继续执行，所以即使场景B非常大也不会感觉到卡顿。

同步附加式载入：如果当前场景是A，我们要载入场景B，unity在载入B的时候不会销毁A，也就是说载入结束之后A和B将同一时候存在，这个适合于世界地图比较大的时候。依据人走到的位置来动态的载入世界地图中的不同部分。当然前提是将世界地图做成多个不同的scene文件。

异步附加式加：这样的情况与上一种基本一样，差别仍然在于载入B的过程是在一个后台线程中异步进行的。

2. 代码过时

a. 新代码

```
1 public class SenceManager : MonoBehaviour
2 {
3     •     public void OnClick()
4     •     {
5         •         SceneManager.LoadScene("所要转换的场景")
6     •     }
7 }
```

b. 旧代码

```
1 public class SenceManager : MonoBehaviour
2 {
3     •     public void OnStartGame(string SampleScene)
4     •     {
5         •         Application.LoadLevel(SampleScene);
6     •     }
7 }
```

脚本代码没有导入到对应的场景之中，所要转载场景在链接时单词书写错误。没有保存与应用到场景中。

II. 素材的收集以及无法导入

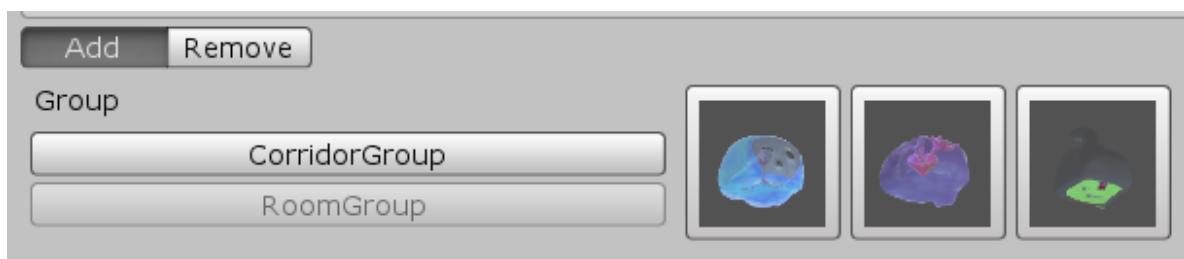
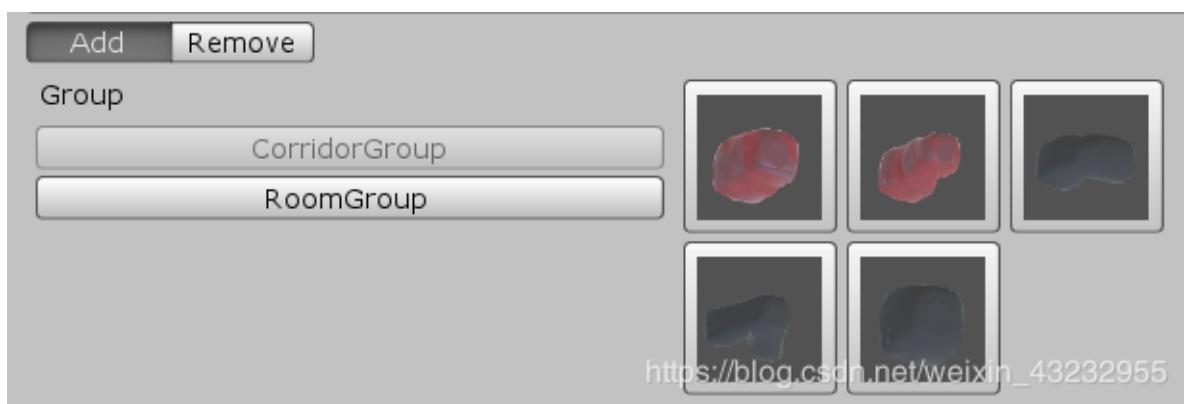
- 开始在网站上收集的素材在Unity导入之后无法使用，即点入测试场景的时候不能进入场景
- 后续在Unity中收集游戏素材和模型，在爱给网寻找零件素材；在Unity的素材名为Creater Kit : FPS，即可直接使用，场景加载问题解决

III. 物理组件和碰撞器的使用

首先是游戏场景中的血管，牙齿，粘膜和骨头等会有穿模的问题，需要添加相应的 `Capsule Colider`，`Mesh Colider`，`Mesh Collider`（网格碰撞体）根据 Mesh 形状产生碰撞体，比起 `Box Collider`、`Sphere Collider` 和 `Capsule Collider`，`Mesh Collider` 更加精确，但会占用更多的系统资源，专门用于复杂网格所生成的模型，添加之后便不会有碰撞器，所以不再会穿模

IV. 地图场景的搭建

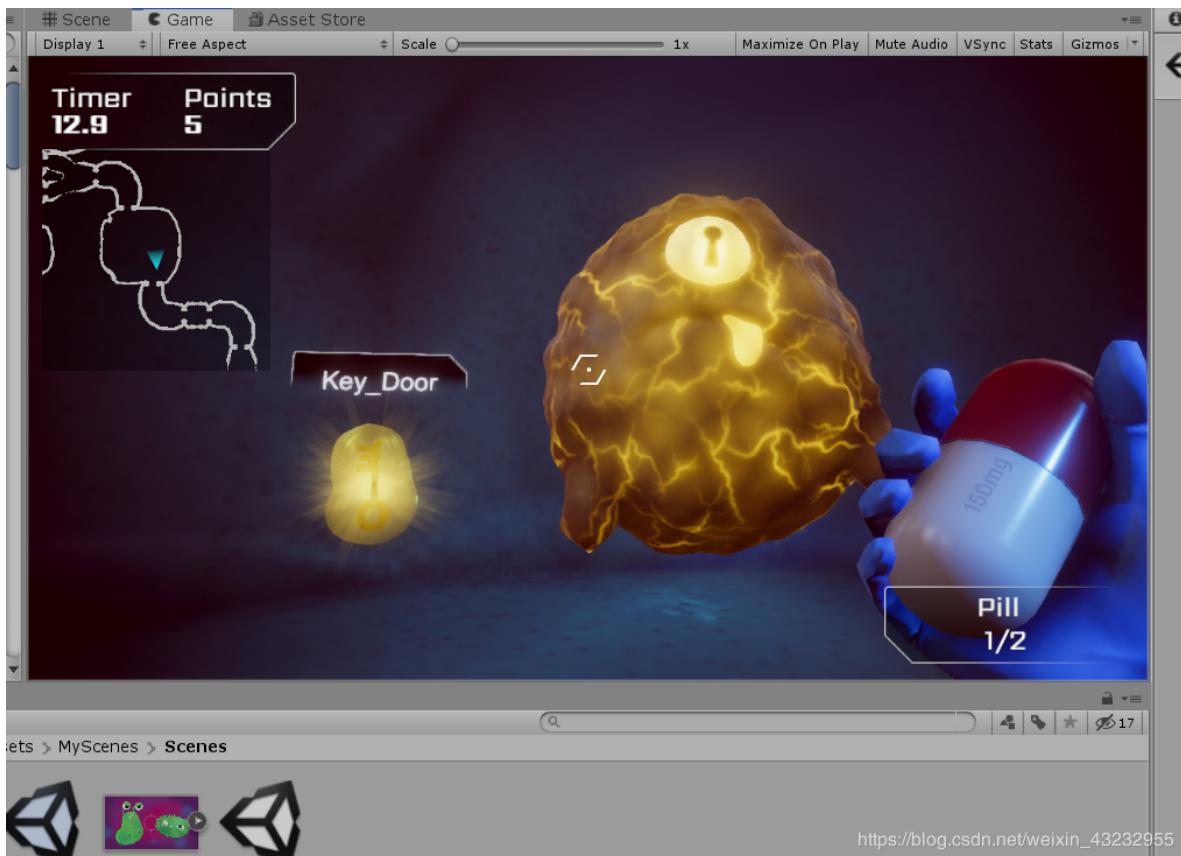
在地图场景搭建时，虽然有各个地图空间的模型，但是先要构思一张有趣的，难度合适的地图，而且每个空间要互相连接并且能够彼此穿梭。



通过多次的测试以及修改，搭建出了本场景的地图

V. 门和钥匙Key - Value的对应

游戏的关键就是通过寻找一个个钥匙，打开各个血管通道的门，从而消灭细菌，抵达终点。所以每一个钥匙和门都是 `Key-Value` 的形式，并不是一个钥匙开启所有的门



需要为钥匙指定名称而不能重复，将其添加到对应的门上。

测试时，出现多次拿到钥匙打不开门，甚至没有钥匙也能开门的情况，需要多次调整修改。

5.2 游戏开发心得

通过本次3D游戏的开发，我们熟练地使用了Unity3D开发工具，并且较为深入地了解了3D游戏的开发过程，学会了在游戏中使用刚体，碰撞，角色控制器等物理引擎，学会对自己游戏场景进行美感上的设计，并且会使用C#脚本语言对游戏进行操控。

更重要的是我们以小组的形式进行开发和创作，在互助中学会了分工合作，各显其长。也在开发3D游戏的过程中了解到了现今3D游戏的发展前景和我国目前在该市场上的优势与不足。

通过本次3D游戏的开发，我们熟练地使用了Unity3D开发工具，并且较为深入地了解了3D游戏的开发过程，学会了在游戏中使用刚体，碰撞，角色控制器等物理引擎，学会对自己游戏场景进行美感上的设计，并且会使用C#脚本语言对游戏进行操控。更重要的是我们以小组的形式进行开发和创作，在互助中学会了分工合作，各显其长。也在开发3D游戏的过程中了解到了现今3D游戏的发展前景和我国目前在该市场上的优势与不足。

本报告从游戏的开始界面、游戏界面、游戏对象几面等几个模块来介绍利用Unity3D引擎并结合C#来开发一款简单的坦克游戏。游戏中包含了几个常用的重点技术，如射线检测、碰撞检测、敌人的AI等。它基本实现了一般的游戏功能，同时它的可扩展性还很高，可以在此基础上另外添加一些比较有创意的设计使该游戏更加的完整。

本系统主要是用Unity3D进行系统整体框架的搭建，但Unity3D对于我来说是比较陌生的，再接触前根本没有听说过，在网上有大量的资料可供查阅，从中提取有用信息的同时也让我对这款软件有了进一步的认识和了解。编程的过程对于我们本身也是一个学习进步的过程，我们自己在编程的水平十分有限，经常在一些简单的地方出错误，因为我只能去网上或者老师来指导我，给了我很大的帮助，最后我

可以独立的完成功能的实现。

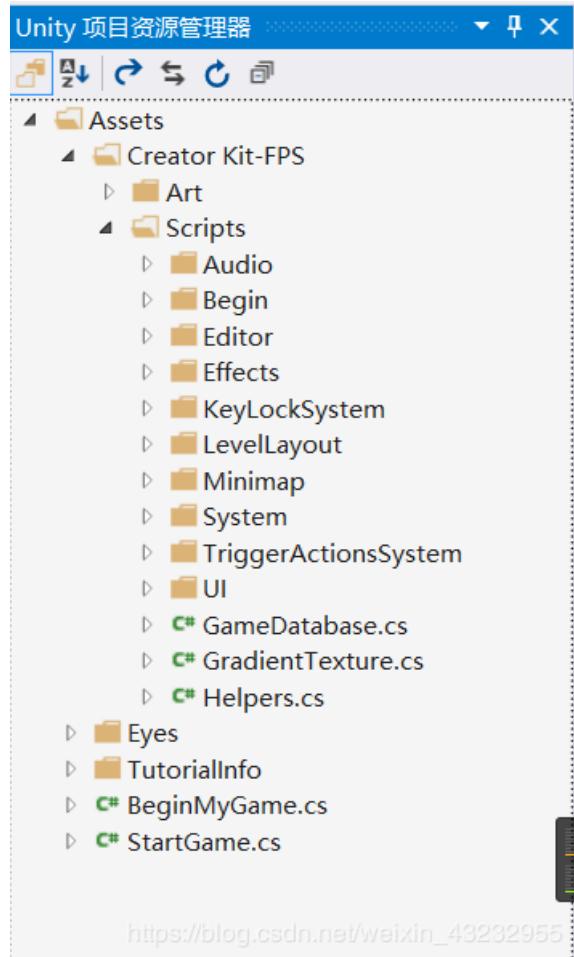
这款游戏有不足之处，可拓展性还有很多，包括关卡的设置以及细菌的伤害，场景的进一步的优化等等问题，先将其部署在Github上，方便后续进一步的优化和开发。

通过本次3D游戏的开发和对Unity3D软件的熟悉，对于我们今后对于3D游戏的进一步开发起着非常大的促进作用。

第七章 游戏源代码

由于模型大部分收集自 [Unity Asset Store](#)，部分来自 [爱给网](#)，模型的逻辑代码均已封装，以下代码是整体的控制逻辑以及控制模型的行为代码

[代码树状图](#)



Scripts

7.1 游戏开始Start UI 场景切换

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class Start : MonoBehaviour
7  {
8      public void OnClick()
9      {
10          SceneManager.LoadScene("MyScene");
11          //Application.LoadLevel(scene);
12      }
13 }
```

7.2 游戏音频

玩家音乐

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
```

```

4
5 [RequireComponent(typeof(AudioSource))]
6 public class RandomPlayer : MonoBehaviour
7 {
8     public AudioClip[] Clips;
9     public float PitchMin = 1.0f;
10    public float PitchMax = 1.0f;
11
12    public AudioSource source => m_Source;
13
14    AudioSource m_Source;
15
16    void Awake()
17    {
18        m_Source = GetComponent<AudioSource>();
19    }
20
21    public AudioClip GetRandomClip()
22    {
23        return Clips[Random.Range(0, Clips.Length)];
24    }
25
26    public void PlayRandom()
27    {
28        if(Clips.Length == 0)
29            return;
30
31        PlayClip(GetRandomClip(), PitchMin, PitchMax);
32    }
33
34    public void PlayClip(AudioClip clip, float pitchMin, float pitchMax)
35    {
36        m_Source.pitch = Random.Range(pitchMin, pitchMax);
37        m_Source.PlayOneShot(clip);
38    }
39}

```

UI菜单音效

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class UIAudioPlayer : MonoBehaviour
6  {
7      public static UIAudioPlayer Instance { get; private set; }
8
9      public AudioClip PositiveSound;
10     public AudioClip NegativeSound;
11
12     AudioSource m_Source;
13
14     void Awake()
15     {
16         m_Source = GetComponent<AudioSource>();
17         Instance = this;
18     }
19

```

```

20     public static void PlayPositive()
21     {
22         Instance.m_Source.PlayOneShot(Instance.PositiveSound);
23     }
24
25     public static void PlayNegative()
26     {
27         Instance.m_Source.PlayOneShot(Instance.NegativeSound);
28     }
29 }
```

7.3 游戏中创建新场景

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Data.OleDb;
4  using UnityEngine;
5  using UnityEditor;
6  using UnityEditor.SceneManagement;
7
8  public class CreateSceneMenu
9  {
10     [MenuItem("FPSKIT/Create new Scene")]
11     static void CreateSceneRoom()
12     {
13         NewScene("Template_room");
14     }
15
16     [MenuItem("FPSKIT/Create new Empty Scene")]
17     static void CreateSceneEmpty()
18     {
19         NewScene("Template");
20     }
21
22     static void NewScene(string originalScene)
23     {
24         string originalPath = "Assets/Scenes/Template.unity";
25         string newPath = EditorUtility.SaveFilePanel("New Scene", "Assets/Creator
Kit - FPS/Scenes", "MyScene", "unity");
26
27         if (!string.IsNullOrEmpty(newPath))
28         {
29             newPath = newPath.Replace(Application.dataPath, "Assets");
30
31             //if return false, maybe they moved the original asset, try to search
32             for it
33                 if (!AssetDatabase.CopyAsset(originalPath, newPath))
34                 {
35                     string[] foundAssets = AssetDatabase.FindAssets($"#{originalScene}
36
37                     if (foundAssets.Length == 0)
38                     {
39                         Debug.LogError("Couldn't find the Template scene. Did you
40 delete it? See the manual for how to create a scene manually");
41                     }
42                     else
```

```

41             {
42                 originalPath = AssetDatabase.GUIDToAssetPath(foundAssets[0]);
43                 if (!AssetDatabase.CopyAsset(originalPath, newPath))
44                 {
45                     Debug.LogErrorFormat("Couldn't copy Template scene at {0}. See the manual for how to create a scene manually", originalPath);
46                 }
47             else
48             {
49                 EditorSceneManager.OpenScene(newPath);
50             }
51         }
52     }
53 }
54 }
55 }
```

```

1  using UnityEditor;
2  using UnityEngine;
3
4  [CustomEditor(typeof(GradientTexture))]
5  public class GradientTextureEditor : Editor
6  {
7      public override void OnInspectorGUI()
8      {
9          using (var cc = new EditorGUI.ChangeCheckScope())
10         {
11             base.OnInspectorGUI();
12             if (cc.changed)
13                 BuildTexture();
14         }
15     }
16 }
17 void BuildTexture()
18 {
19     var a = target as GradientTexture;
20     if (a.texture == null || a.texture.width != a.width)
21     {
22         if (a.texture != null) DestroyImmediate(a.texture, true);
23         a.texture = new Texture2D(a.width, 4, TextureFormat.ARGB32, true);
24         a.texture.name = a.name;
25         AssetDatabase.AddObjectToAsset(a.texture, a);
26     }
27     var pixels = a.texture.GetPixels();
28     for (var x = 0; x < a.texture.width; x++)
29     {
30         var color = a.gradient.Evaluate(1f * x / a.texture.width);
31         for (var y = 0; y < 4; y++)
32         {
33             pixels[y * a.texture.width + x] = color;
34         }
35     }
36     a.texture.SetPixels(pixels);
37     a.texture.Apply();
38     a.texture.wrapMode = TextureWrapMode.Clamp;
39     EditorUtility.SetDirty(target);
```

```

40         AssetDatabase.SaveAssets();
41     }
42 }

1  using UnityEditor;
2  using UnityEditor.SceneManagement;
3  using UnityEngine;
4
5  [InitializeOnLoad]
6  public static class HideSkybox
7  {
8      static HideSkybox ()
9      {
10         EditorApplication.update += FirstFrameUpdate;
11     }
12
13     static void FirstFrameUpdate ()
14     {
15         bool skyboxUpdated = UpdateSceneViewSkybox ();
16
17         if(skyboxUpdated)
18             EditorApplication.update -= FirstFrameUpdate;
19     }
20
21     static bool UpdateSceneViewSkybox ()
22     {
23         SceneView sceneView = SceneView.lastActiveSceneView;
24
25         if (sceneView == null)
26             return false;
27
28         SceneView.SceneViewState state = sceneView.sceneViewState;
29         state.SetAllEnabled(false);
30         sceneView.sceneViewState = state;
31
32         return true;
33     }
34 }

```

7.4 第一人称视角及特效

相机跟随

```

1  using UnityEditor;
2  using UnityEditor.SceneManagement;
3  using UnityEngine;
4
5  [InitializeOnLoad]
6  public static class HideSkybox
7  {
8      static HideSkybox ()
9      {
10         EditorApplication.update += FirstFrameUpdate;
11     }
12

```

```

13     static void FirstFrameUpdate ()
14     {
15         bool skyboxUpdated = UpdateSceneViewSkybox ();
16
17         if(skyboxUpdated)
18             EditorApplication.update -= FirstFrameUpdate;
19     }
20
21     static bool UpdateSceneViewSkybox ()
22     {
23         SceneView sceneView = SceneView.lastActiveSceneView;
24
25         if (sceneView == null)
26             return false;
27
28         SceneView.SceneViewState state = sceneView.sceneViewState;
29         state.SetAllEnabled(false);
30         sceneView.sceneViewState = state;
31
32         return true;
33     }
34 }
```

子弹发射粒子效果

```

1  using System.Collections.Generic;
2  using UnityEngine;
3
4  /// <summary>
5  /// This handle impacts on object from the raycast of the weapon. It will create
6  /// a pool of the prefabs for performance
7  /// optimisation.
8  /// </summary>
9  public class ImpactManager : MonoBehaviour
10 {
11     [System.Serializable]
12     public class ImpactSetting
13     {
14         public ParticleSystem ParticlePrefab;
15         public AudioClip ImpactSound;
16         public Material TargetMaterial;
17     }
18
19     static public ImpactManager Instance { get; protected set; }
20
21     public ImpactSetting DefaultSettings;
22     public ImpactSetting[] ImpactSettings;
23
24     Dictionary<Material, ImpactSetting> m_SettingLookup = new
25     Dictionary<Material, ImpactSetting>();
26
27     // Start is called before the first frame update
28     void Awake()
29     {
30         Instance = this;
31     }
32 }
```

```

30
31     void Start()
32     {
33         PoolSystem.Instance.InitPool(DefaultSettings.ParticlePrefab, 32);
34         foreach (var impactSettings in ImpactSettings)
35         {
36             PoolSystem.Instance.InitPool(impactSettings.ParticlePrefab, 32);
37             m_SettingLookup.Add(impactSettings.TargetMaterial, impactSettings);
38         }
39     }
40
41     public void PlayImpact(Vector3 position, Vector3 normal, Material material =
42     null)
43     {
44         ImpactSetting setting = null;
45         if (material == null || !m_SettingLookup.TryGetValue(material, out
46         setting))
47         {
48             setting = DefaultSettings;
49         }
50
51         var sys = PoolSystem.Instance.GetInstance<ParticleSystem>
52         (setting.ParticlePrefab);
53         sys.gameObject.transform.position = position;
54         sys.gameObject.transform.forward = normal;
55
56         var source = WorldAudioPool.GetWorldSFXSource();
57
58         source.transform.position = position;
59         source.pitch = Random.Range(0.8f, 1.1f);
60         source.PlayOneShot(setting.ImpactSound);
61     }
62 }
```

消灭细菌时的液体特效

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class LiquidAmmoDisplay : AmmoDisplay
6  {
7      public LiquidContainer Container;
8      public float MinLiquidAmount;
9      public float MaxLiquidAmount;
10
11     public override void UpdateAmount(int current, int max)
12     {
13         Container.ChangeLiquidAmount(Mathf.Lerp(MinLiquidAmount, MaxLiquidAmount,
14         current/(float)max));
15     }
16 }
```

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class TextAmmoDisplay : AmmoDisplay
7  {
8      public Text DisplayText;
9
10     public override void UpdateAmount(int current, int max)
11     {
12         DisplayText.text = current.ToString();
13     }
14 }
```

7.5 钥匙和锁

Key

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class Key : MonoBehaviour
7  {
8      public string keyType;
9      public Text KeyNameText;
10
11     void OnEnable()
12     {
13         KeyNameText.text = keyType;
14     }
15
16     void OnTriggerEnter(Collider other)
17     {
18         var keychain = other.GetComponent<Keychain>();
19
20         if (keychain != null)
21         {
22             keychain.GrabbedKey(keyType);
23             Destroy(gameObject);
24         }
25     }
26 }
```

Lock

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  #if UNITY_EDITOR
6  using UnityEditor;
7  #endif
8
```

```

9  public class Lock : GameTrigger
10 {
11     public string keyType;
12     public Text KeyNameText;
13
14     Canvas m_Canvas;
15
16     void Start()
17     {
18         KeyNameText.text = keyType;
19
20         m_Canvas = KeyNameText.GetComponentInParent<Canvas>();
21         m_Canvas.gameObject.SetActive(false);
22     }
23
24
25     void OnTriggerEnter(Collider other)
26     {
27         m_Canvas.gameObject.SetActive(true);
28
29         var keychain = other.GetComponent<Keychain>();
30
31         if (keychain != null && keychain.HaveKey(keyType))
32         {
33             keychain.UseKey(keyType);
34             Opened();
35             //just destroy the script, if it's on the door we don't want to
36             //destroy the door.
37             Destroy(this);
38             Destroy(m_Canvas.gameObject);
39         }
40     }
41
42     void OnTriggerExit(Collider other)
43     {
44         m_Canvas.gameObject.SetActive(false);
45     }
46
47     public virtual void Opened()
48     {
49         Trigger();
50     }
51
52 #if UNITY_EDITOR
53 [CustomEditor(typeof(Lock))]
54 public class LockEditor : Editor
55 {
56     SerializedProperty m_ActionListProperty;
57     SerializedProperty m_KeyNameTextProperty;
58     Lock m_Lock;
59
60     int m_KeyTypeIndex = -1;
61     string[] m_AllKeyType = new string[0];
62
63     void OnEnable()
64     {
65         m_Lock = target as Lock;

```

```

66     m_ActionListProperty = serializedObject.FindProperty("actions");
67     m_KeyNameTextProperty = serializedObject.FindProperty("KeyNameText");
68
69     var allKeys = Resources.FindObjectsOfTypeAll<Key>();
70     foreach (var key in allKeys)
71     {
72         ArrayUtility.Add(ref m_AllKeyType, key.keyType);
73
74         if (m_Lock.keyType == key.keyType)
75         {
76             m_KeyTypeIDex = m_AllKeyType.Length - 1;
77         }
78     }
79 }
80
81 public override void OnInspectorGUI()
82 {
83     EditorGUILayout.PropertyField(m_KeyNameTextProperty);
84     EditorGUILayout.PropertyField(m_ActionListProperty, true);
85
86     if (m_AllKeyType.Length > 0)
87     {
88         int index = EditorGUILayout.Popup("Key Type", m_KeyTypeIDex,
89         m_AllKeyType);
90         if (index != m_KeyTypeIDex)
91         {
92             Undo.RecordObject(m_Lock, "Changed Key Type");
93
94             m_Lock.keyType = m_AllKeyType[index];
95             m_KeyTypeIDex = index;
96         }
97     }
98     else
99     {
100        EditorGUILayout.HelpBox("Add at least a key in the scene to be able
101        to select the type here", MessageType.Warning);
102    }
103
104 }
105 #endif

```

Key-Value 开门

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Keychain: MonoBehaviour
6  {
7      List<string> m_KeyTypeOwned = new List<string>();
8
9      public void GrabbedKey(string keyType)
10     {
11         m_KeyTypeOwned.Add(keyType);
12     }
13

```

```

14     public bool HaveKey(string keyType)
15     {
16         return m_KeyTypeOwned.Contains(keyType);
17     }
18
19     public void UseKey(string keyType)
20     {
21         m_KeyTypeOwned.Remove(keyType);
22     }
23 }
```

7.6 地图场景和主场景部分代码

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Linq;
4  using UnityEngine;
5  #if UNITY_EDITOR
6  using UnityEditor;
7  #endif
8
9  [ExecuteInEditMode]
10 public class LevelRoom : MonoBehaviour
11 {
12     public Transform[] Exits;
13
14     [HideInInspector]
15     public LevelRoom[] ExitDestination;
16
17     [HideInInspector]
18     public LevelLayout Owner;
19
20     /// <summary>
21     /// Called by the editor script that place room to initialize the ExitUsed
22     /// array to match Exits
23     public void Placed(LevelLayout layoutOwner)
24     {
25         Owner = layoutOwner;
26         ExitDestination = new LevelRoom[Exits.Length];
27     }
28 #if UNITY_EDITOR
29     public void Removed()
30     {
31         if (ExitDestination != null)
32         {
33             for (int i = 0; i < ExitDestination.Length; ++i)
34             {
35                 if (ExitDestination[i] != null)
36                 {
37                     SerializedObject otherObj = new
38                     SerializedObject(ExitDestination[i]);
39                     var connectorProp =
otherObj.FindProperty(nameof(ExitDestination));
40             }
41         }
42     }
43 }
```

```

40             for (int k = 0; k < connectorProp.arraySize; ++k)
41             {
42                 var prop = connectorProp.GetArrayElementAtIndex(k);
43
44                 if (prop.objectReferenceValue == this)
45                 {
46                     prop.objectReferenceValue = null;
47                     prop.serializedObject.ApplyModifiedProperties();
48                 }
49             }
50         }
51     }
52 }
53
54 if (Owner != null && !Owner.Destroyed)
55 {
56     SerializedObject ownerObject = new SerializedObject(Owner);
57
58     var piecesProp = ownerObject.FindProperty(nameof(Owner.rooms));
59
60     for (int i = 0; i < piecesProp.arraySize; ++i)
61     {
62         var prop = piecesProp.GetArrayElementAtIndex(i);
63
64         if (prop.objectReferenceValue == this)
65         {
66             piecesProp.DeleteArrayElementAtIndex(i);
67             piecesProp.DeleteArrayElementAtIndex(i);
68             break;
69         }
70     }
71     ownerObject.ApplyModifiedProperties();
72 }
73 }
74 }
75 #endif
76 }
```

7.7 MiniMap

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Rendering;
5
6  /// 这个脚本附加到任何需要在小地图中渲染的渲染器对象上
7  public class MinimapElement : MonoBehaviour
8  {
9      public static List<Renderer> Renderers => s_Renderers;
10
11     static List<Renderer> s_Renderers = new List<Renderer>();
12
13     Renderer m_Renderer;
14
15     void OnEnable()
```

```

16     {
17         m_Renderer = GetComponent<Renderer>();
18
19         if(m_Renderer != null)
20             s_Renderers.Add(m_Renderer);
21     }
22
23     void OnDisable()
24     {
25         if (m_Renderer)
26             s_Renderers.Remove(m_Renderer);
27     }
28 }
```

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Numerics;
4  using UnityEngine;
5  using UnityEngine.Rendering;
6  using Matrix4x4 = UnityEngine.Matrix4x4;
7  using Quaternion = UnityEngine.Quaternion;
8  using Vector3 = UnityEngine.Vector3;
9
10 /// <summary>
11 /// Minimap System that render the level into a given render texture with the
12 /// given settings. It rely on the level part
13 /// that need to be on the minimap (e.g. most of the time the wall) to have a
14 /// MinimapElement script attached to it.
15 /// It is used by the Minimap UI script to render the minimap.
16 /// </summary>
17 public class MinimapSystem
18 {
19     [System.Serializable]
20     public struct MinimapSystemSetting
21     {
22         public float heightStep;
23         public float halfSize;
24         public float wallThickness;
25         public Material material;
26         public bool isFixed;
27     }
28
29     static int s_WallThicknessId;
30
31     static MinimapSystem()
32     {
33         s_WallThicknessId = Shader.PropertyToID("_WallThickness");
34     }
35
36     public static void Render(RenderTexture renderTarget, Vector3 origin, Vector3
37 forward, MinimapSystemSetting settings)
38     {
39         settings.material.SetFloat(s_WallThicknessId, settings.wallThickness);
40
41         float aspectRatio = renderTarget.width / (float)renderTarget.height;
```

```

41         CommandBuffer buffer = new CommandBuffer();
42
43         Matrix4x4 lookAt;
44
45         Vector3 camPos = origin + Vector3.up * 3.0f;
46
47         if (settings.isFixed)
48         {
49             lookAt = Matrix4x4.TRS(camPos, Quaternion.LookRotation(Vector3.down,
50             Vector3.forward), new Vector3(1, 1, -1).inverse;
51         }
52         else
53         {
54             lookAt = Matrix4x4.TRS(camPos, Quaternion.LookRotation(Vector3.down,
55             forward), new Vector3(1, 1, -1).inverse;
56         }
57
58         buffer.SetRenderTarget(renderTarget);
59         buffer.SetProjectionMatrix(Matrix4x4.Ortho(-settings.halfSize *
60             aspectRatio, settings.halfSize * aspectRatio, -settings.halfSize ,
61             settings.halfSize , 0.5f, 1.5f));
62         buffer.SetViewMatrix(lookAt);
63
64         buffer.ClearRenderTarget(true, true, Color.black);
65         foreach (var r in MinimapElement.Renderers)
66         {
67             buffer.DrawRenderer(r, settings.material);
68         }
69     }
70 }
```

7.8 武器

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Linq;
4  using UnityEngine;
5  using UnityEngine.Serialization;
6  using UnityEngine.UI;
7  #if UNITY_EDITOR
8  using UnityEditor;
9  #endif
10
11 public class Weapon : MonoBehaviour
12 {
13     static RaycastHit[] s_HitInfoBuffer = new RaycastHit[8];
14
15     public enum TriggerType
16     {
17         Auto,
18         Manual
19     }
20 }
```

```
21     public enum WeaponType
22     {
23         Raycast,
24         Projectile
25     }
26
27     public enum WeaponState
28     {
29         Idle,
30         Firing,
31         Reloading
32     }
33
34     [System.Serializable]
35     public class AdvancedSettings
36     {
37         public float spreadAngle = 0.0f;
38         public int projectilePerShot = 1;
39         public float screenShakeMultiplier = 1.0f;
40     }
41
42     public TriggerType triggerType = TriggerType.Manual;
43     public WeaponType weaponType = WeaponType.Raycast;
44     public float fireRate = 0.5f;
45     public float reloadTime = 2.0f;
46     public int clipSize = 4;
47     public float damage = 1.0f;
48
49     [AmmoType]
50     public int ammoType = -1;
51
52     public Projectile projectilePrefab;
53     public float projectileLaunchForce = 200.0f;
54
55     public Transform EndPoint;
56
57     public AdvancedSettings advancedSettings;
58
59     [Header("Animation Clips")]
60     public AnimationClip FireAnimationClip;
61     public AnimationClip ReloadAnimationClip;
62
63     [Header("Audio Clips")]
64     public AudioClip FireAudioClip;
65     public AudioClip ReloadAudioClip;
66
67     [Header("Visual Settings")]
68     public LineRenderer PrefabRayTrail;
69
70     [Header("Visual Display")]
71     public AmmoDisplay AmmoDisplay;
72
73     public bool triggerDown
74     {
75         get { return m_TriggerDown; }
76         set
77         {
78             m_TriggerDown = value;
```

```

79             if (!m_TriggerDown) m_ShotDone = false;
80         }
81     }
82
83     public WeaponState CurrentState => m_CurrentState;
84     public int ClipContent => m_ClipContent;
85     public Controller Owner => m_Owner;
86
87     Controller m_Owner;
88
89     Animator m_Animator;
90     WeaponState m_CurrentState;
91     bool m_ShotDone;
92     float m_ShotTimer = -1.0f;
93     bool m_TriggerDown;
94     int m_ClipContent;
95
96     AudioSource m_Source;
97
98     Vector3 m_ConvertedMuzzlePos;
99
100    class ActiveTrail
101    {
102        public LineRenderer renderer;
103        public Vector3 direction;
104        public float remainingTime;
105    }
106
107    List<ActiveTrail> m_ActiveTrails = new List<ActiveTrail>();
108
109    Queue<Projectile> m_ProjectilePool = new Queue<Projectile>();
110
111    int fireNameHash = Animator.StringToHash("fire");
112    int reloadNameHash = Animator.StringToHash("reload");
113
114    void Awake()
115    {
116        m_Animator = GetComponentInChildren<Animator>();
117        m_Source = GetComponentInChildren<AudioSource>();
118        m_ClipContent = clipSize;
119
120        if (PrefabRayTrail != null)
121        {
122            const int trailPoolSize = 16;
123            PoolSystem.Instance.InitPool(PrefabRayTrail, trailPoolSize);
124        }
125
126        if (projectilePrefab != null)
127        {
128            //a minimum of 4 is useful for weapon that have a clip size of 1 and
129            //where you can throw a second
130            //or more before the previous one was recycled/exploded.
131            int size = Mathf.Max(4, clipSize) *
132                advancedSettings.projectilePerShot;
133            for (int i = 0; i < size; ++i)
134            {
135                Projectile p = Instantiate(projectilePrefab);
136                p.gameObject.SetActive(false);

```

```

135             m_ProjectilePool.Enqueue(p);
136         }
137     }
138 }
139
140 public void PickedUp(Controller c)
141 {
142     m_Owner = c;
143 }
144
145 public void PutAway()
146 {
147     m_Animator.WriteDefaultValues();
148
149     for (int i = 0; i < m_ActiveTrails.Count; ++i)
150     {
151         var activeTrail = m_ActiveTrails[i];
152         m_ActiveTrails[i].renderer.gameObject.SetActive(false);
153     }
154
155     m_ActiveTrails.Clear();
156 }
157
158 public void Selected()
159 {
160     var ammoRemaining = m_Owner.GetAmmo(ammoType);
161
162     //gun get disabled when ammo is == 0 and there is no more ammo in the
163     //clip, so this allow to re-enable it if we
164     //grabbed ammo since last time we switched
165     gameObject.SetActive(ammoRemaining != 0 || m_ClipContent != 0);
166
167     if(FireAnimationClip != null)
168         m_Animator.SetFloat("fireSpeed", FireAnimationClip.length /
169         fireRate);
170
171     if(ReloadAnimationClip != null)
172         m_Animator.SetFloat("reloadSpeed", ReloadAnimationClip.length /
173         reloadTime);
174
175     m_CurrentState = WeaponState.Idle;
176
177     triggerDown = false;
178     m_ShotDone = false;
179
180     WeaponInfoUI.Instance.UpdateWeaponName(this);
181     WeaponInfoUI.Instance.UpdateClipInfo(this);
182     WeaponInfoUI.Instance.UpdateAmmoAmount(m_Owner.GetAmmo(ammoType));
183
184     if(AmmoDisplay)
185         AmmoDisplay.UpdateAmount(m_ClipContent, clipSize);
186
187     if (m_ClipContent == 0 && ammoRemaining != 0)
188     {
189         //this can only happen if the weapon ammo reserve was empty and we
190         //picked some since then. So directly
191         //reload the clip when weapon is selected
192         int chargeInClip = Mathf.Min(ammoRemaining, clipSize);

```

```

189         m_ClipContent += chargeInClip;
190         if(AmmoDisplay)
191             AmmoDisplay.UpdateAmount(m_ClipContent, clipSize);
192         m_Owner.ChangeAmmo(ammoType, -chargeInClip);
193         WeaponInfoUI.Instance.UpdateClipInfo(this);
194     }
195
196     m_Animator.SetTrigger("selected");
197 }
198
199 public void Fire()
200 {
201     if (m_CurrentState != WeaponState.Idle || m_ShotTimer > 0 || 
202     m_ClipContent == 0)
203         return;
204
205     m_ClipContent -= 1;
206
207     m_ShotTimer = fireRate;
208
209     if(AmmoDisplay)
210         AmmoDisplay.UpdateAmount(m_ClipContent, clipSize);
211
212     WeaponInfoUI.Instance.UpdateClipInfo(this);
213
214     //the state will only change next frame, so we set it right now.
215     m_CurrentState = WeaponState.Firing;
216
217     m_Animator.SetTrigger("fire");
218
219     m_Source.pitch = Random.Range(0.7f, 1.0f);
220     m_Source.PlayOneShot(FireAudioClip);
221
222     CameraShaker.Instance.Shake(0.2f, 0.05f *
223     advancedSettings.screenShakeMultiplier);
224
225     if (weaponType == WeaponType.Raycast)
226     {
227         for (int i = 0; i < advancedSettings.projectilePerShot; ++i)
228         {
229             RaycastShot();
230         }
231     }
232     else
233     {
234         ProjectileShot();
235     }
236 }
237
238 void RaycastShot()
239 {
240
241     //compute the ratio of our spread angle over the fov to know in viewport
242     //space what is the possible offset from center
243     float spreadRatio = advancedSettings.spreadAngle /
244     Controller.Instance.MainCamera.fieldOfView;
245 }
```

```

243         Vector2 spread = spreadRatio * Random.insideUnitCircle;
244
245         RaycastHit hit;
246         Ray r = Controller.Instance.MainCamera.ViewportPointToRay(Vector3.one *
247             0.5f + (Vector3)spread);
248         Vector3 hitPosition = r.origin + r.direction * 200.0f;
249
250         if (Physics.Raycast(r, out hit, 1000.0f, ~(1 << 9),
251             QueryTriggerInteraction.Ignore))
252         {
253             Renderer renderer = hit.collider.GetComponentInChildren<Renderer>();
254             ImpactManager.Instance.PlayImpact(hit.point, hit.normal, renderer ==
255             null ? null : renderer.sharedMaterial);
256
257             //if too close, the trail effect would look weird if it arced to hit
258             //the wall, so only correct it if far
259             if (hit.distance > 5.0f)
260                 hitPosition = hit.point;
261
262             //this is a target
263             if (hit.collider.gameObject.layer == 10)
264             {
265                 Target target = hit.collider.gameObject.GetComponent<Target>();
266                 target.Got(damage);
267             }
268         }
269
270         if (PrefabRayTrail != null)
271         {
272             var pos = new Vector3[] { GetCorrectedMuzzlePlace(), hitPosition };
273             var trail = PoolSystem.Instance.GetInstance<LineRenderer>
274             (PrefabRayTrail);
275             trail.gameObject.SetActive(true);
276             trail.SetPositions(pos);
277             m_ActiveTrails.Add(new ActiveTrail()
278             {
279                 remainingTime = 0.3f,
280                 direction = (pos[1] - pos[0]).normalized,
281                 renderer = trail
282             });
283         }
284
285         void ProjectileShot()
286         {
287             for (int i = 0; i < advancedSettings.projectilePerShot; ++i)
288             {
289                 float angle = Random.Range(0.0f, advancedSettings.spreadAngle *
290                     0.5f);
291                 Vector2 angleDir = Random.insideUnitCircle * Mathf.Tan(angle *
292                     Mathf.Deg2Rad);
293
294                 Vector3 dir = EndPoint.transform.forward + (Vector3)angleDir;
295                 dir.Normalize();
296
297                 var p = m_ProjectilePool.Dequeue();

```

```
294             p.gameObject.SetActive(true);
295             p.Launch(this, dir, projectileLaunchForce);
296         }
297     }
298
299     //For optimization, when a projectile is "destroyed" it is instead disabled
300     //and return to the weapon for reuse.
301     public void ReturnProjectile(Projectile p)
302     {
303         m_ProjectilePool.Enqueue(p);
304     }
305
306     public void Reload()
307     {
308         if (m_CurrentState != WeaponState.Idle || m_ClipContent == clipSize)
309             return;
310
311         int remainingBullet = m_Owner.GetAmmo(ammoType);
312
313         if (remainingBullet == 0)
314         {
315             //No more bullet, so we disable the gun so it's not displayed
316             //anymore and change weapon
317             gameObject.SetActive(false);
318             return;
319         }
320
321         if (ReloadAudioClip != null)
322         {
323             m_Source.pitch = Random.Range(0.7f, 1.0f);
324             m_Source.PlayOneShot(ReloadAudioClip);
325         }
326
327         int chargeInClip = Mathf.Min(remainingBullet, clipSize - m_ClipContent);
328
329         //the state will only change next frame, so we set it right now.
330         m_CurrentState = WeaponState.Reload;
331
332         m_ClipContent += chargeInClip;
333
334         if(AmmoDisplay)
335             AmmoDisplay.UpdateAmount(m_ClipContent, clipSize);
336
337         m_Animator.SetTrigger("reload");
338
339         m_Owner.ChangeAmmo(ammoType, -chargeInClip);
340
341         WeaponInfoUI.Instance.UpdateClipInfo(this);
342     }
343
344     void Update()
345     {
346         UpdateControllerState();
347
348         if (m_ShotTimer > 0)
349             m_ShotTimer -= Time.deltaTime;
350     }
351 }
```

```

350     Vector3[] pos = new Vector3[2];
351     for (int i = 0; i < m_ActiveTrails.Count; ++i)
352     {
353         var activeTrail = m_ActiveTrails[i];
354
355         activeTrail.renderer.GetPositions(pos);
356         activeTrail.remainingTime -= Time.deltaTime;
357
358         pos[0] += activeTrail.direction * 50.0f * Time.deltaTime;
359         pos[1] += activeTrail.direction * 50.0f * Time.deltaTime;
360
361         m_ActiveTrails[i].renderer.SetPositions(pos);
362
363         if (m_ActiveTrails[i].remainingTime <= 0.0f)
364         {
365             m_ActiveTrails[i].renderer.gameObject.SetActive(false);
366             m_ActiveTrails.RemoveAt(i);
367             i--;
368         }
369     }
370 }
371
372 void UpdateControllerState()
373 {
374     m_Animator.SetFloat("speed", m_Owner.Speed);
375     m_Animator.SetBool("grounded", m_Owner.Grounded);
376
377     var info = m_Animator.GetCurrentAnimatorStateInfo(0);
378
379     WeaponState newState;
380     if (info.shortNameHash == fireNameHash)
381         newState = WeaponState.Firing;
382     else if (info.shortNameHash == reloadNameHash)
383         newState = WeaponState.Reload;
384     else
385         newState = WeaponState.Idle;
386
387     if (newState != m_CurrentState)
388     {
389         var oldState = m_CurrentState;
390         m_CurrentState = newState;
391
392         if (oldState == WeaponState.Firing)
393             {//we just finished firing, so check if we need to auto reload
394                 if(m_ClipContent == 0)
395                     Reload();
396             }
397     }
398
399     if (triggerDown)
400     {
401         if (triggerType == TriggerType.Manual)
402         {
403             if (!m_ShotDone)
404             {
405                 m_ShotDone = true;
406                 Fire();
407             }

```

```

408         }
409     else
410         Fire();
411     }
412 }
413
414 /// <summary>
415     /// This will compute the corrected position of the muzzle flash in world
416     /// space. Since the weapon camera use a
417     /// different FOV than the main camera, using the muzzle spot to spawn thing
418     /// rendered by the main camera will appear
419     /// disconnected from the muzzle flash. So this convert the muzzle post from
420     /// world -> view weapon -> clip weapon -> inverse clip main cam -> inverse
421     /// view cam -> corrected world pos
422     /// </summary>
423     /// <returns></returns>
424     public Vector3 GetCorrectedMuzzlePlace()
425     {
426         Vector3 position = EndPoint.position;
427
428         position =
429             Controller.Instance.WeaponCamera.WorldToScreenPoint(position);
430         position = Controller.Instance.MainCamera.ScreenToWorldPoint(position);
431
432         return position;
433     }
434 }
435
436
437 public abstract class AmmoDisplay : MonoBehaviour
438 {
439     public abstract void UpdateAmount(int current, int max);
440 }
441
442 #if UNITY_EDITOR
443
444
445 [CustomPropertyDrawer(typeof(AmmoTypeAttribute))]
446 public class AmmoTypeDrawer : PropertyDrawer
447 {
448     public override void OnGUI(Rect position, SerializedProperty property,
449     GUIContent label)
450     {
451         AmmoDatabase ammoDB = GameDatabase.Instance.ammoDatabase;
452
453         if (ammoDB.entries == null || ammoDB.entries.Length == 0)
454         {
455             EditorGUI.HelpBox(position, "Please define at least 1 ammo type in
456             the Game Database", MessageType.Error);
457         }
458         else
459         {
460             int currentID = property.intValue;
461             int currentIdx = -1;

```

```

460
461             //this is pretty ineffective, maybe find a way to cache that if
462             //prove to take too much time
463             string[] names = new string[ammoDB.entries.Length];
464             for (int i = 0; i < ammoDB.entries.Length; ++i)
465             {
466                 names[i] = ammoDB.entries[i].name;
467                 if (ammoDB.entries[i].id == currentID)
468                     currentIdx = i;
469             }
470             EditorGUI.BeginChangeCheck();
471             int idx = EditorGUI.Popup(position, "Ammo Type", currentIdx, names);
472             if (EditorGUI.EndChangeCheck())
473             {
474                 property.intValue = ammoDB.entries[idx].id;
475             }
476         }
477     }
478 }
479
480 [CustomEditor(typeof(Weapon))]
481 public class WeaponEditor : Editor
482 {
483     SerializedProperty m_TriggerTypeProp;
484     SerializedProperty m_WeaponTypeProp;
485     SerializedProperty m_FireRateProp;
486     SerializedProperty m_ReloadTimeProp;
487     SerializedProperty m_ClipSizeProp;
488     SerializedProperty m_DamageProp;
489     SerializedProperty m_AmmoTypeProp;
490     SerializedProperty m_ProjectilePrefabProp;
491     SerializedProperty m_ProjectileLaunchForceProp;
492     SerializedProperty m_EndPointProp;
493     SerializedProperty m_AdvancedSettingsProp;
494     SerializedProperty m_FireAnimationClipProp;
495     SerializedProperty m_ReloadAnimationClipProp;
496     SerializedProperty m_FireAudioClipProp;
497     SerializedProperty m_ReloadAudioClipProp;
498     SerializedProperty m_PrefabRayTrailProp;
499     SerializedProperty m_AmmoDisplayProp;
500
501     void OnEnable()
502     {
503         m_TriggerTypeProp = serializedObject.FindProperty("triggerType");
504         m_WeaponTypeProp = serializedObject.FindProperty("weaponType");
505         m_FireRateProp = serializedObject.FindProperty("fireRate");
506         m_ReloadTimeProp = serializedObject.FindProperty("reloadTime");
507         m_ClipSizeProp = serializedObject.FindProperty("clipSize");
508         m_DamageProp = serializedObject.FindProperty("damage");
509         m_AmmoTypeProp = serializedObject.FindProperty("ammoType");
510         m_ProjectilePrefabProp =
511             serializedObject.FindProperty("projectilePrefab");
512         m_ProjectileLaunchForceProp =
513             serializedObject.FindProperty("projectileLaunchForce");
514         m_EndPointProp = serializedObject.FindProperty("EndPoint");
515         m_AdvancedSettingsProp =
516             serializedObject.FindProperty("advancedSettings");

```

```

514         m_FireAnimationClipProp =
515             serializedObject.FindProperty("FireAnimationClip");
516         m_ReloadAnimationClipProp =
517             serializedObject.FindProperty("ReloadAnimationClip");
518         m_FireAudioClipProp = serializedObject.FindProperty("FireAudioClip");
519         m_ReloadAudioClipProp = serializedObject.FindProperty("ReloadAudioClip");
520         m_PrefabRayTrailProp = serializedObject.FindProperty("PrefabRayTrail");
521         m_AmmoDisplayProp = serializedObject.FindProperty("AmmoDisplay");
522     }
523
524     public override void OnInspectorGUI()
525     {
526         serializedObject.Update();
527
528         EditorGUILayout.PropertyField(m_TriggerTypeProp);
529         EditorGUILayout.PropertyField(m_WeaponTypeProp);
530         EditorGUILayout.PropertyField(m_FireRateProp);
531         EditorGUILayout.PropertyField(m_ReloadTimeProp);
532         EditorGUILayout.PropertyField(m_ClipSizeProp);
533         EditorGUILayout.PropertyField(m_DamageProp);
534         EditorGUILayout.PropertyField(m_AmmoTypeProp);
535
536         if (m_WeaponTypeProp.intValue == (int)Weapon.WeaponType.Projectile)
537         {
538             EditorGUILayout.PropertyField(m_ProjectilePrefabProp);
539             EditorGUILayout.PropertyField(m_ProjectileLaunchForceProp);
540
541             EditorGUILayout.PropertyField(m_EndPointProp);
542             EditorGUILayout.PropertyField(m_AdvancedSettingsProp, new
543                 GUIContent("Advance Settings"), true);
544             EditorGUILayout.PropertyField(m_FireAnimationClipProp);
545             EditorGUILayout.PropertyField(m_ReloadAnimationClipProp);
546             EditorGUILayout.PropertyField(m_FireAudioClipProp);
547             EditorGUILayout.PropertyField(m_ReloadAudioClipProp);
548
549             if (m_WeaponTypeProp.intValue == (int)Weapon.WeaponType.Raycast)
550             {
551                 EditorGUILayout.PropertyField(m_PrefabRayTrailProp);
552
553                 EditorGUILayout.PropertyField(m_AmmoDisplayProp);
554
555             }
556         }
557     #endif

```

武器切换

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class WeaponAnimationEventHandler : MonoBehaviour
6  {
7      Weapon m_Owner;
8

```

```

9     void Awake()
10    {
11        m_Owner = GetComponentInParent<Weapon>();
12    }
13
14    public void PlayFootstep()
15    {
16        m_Owner.Owner.PlayFootstep();
17    }
18 }
```

7.8 游戏主逻辑

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEditor;
4  using UnityEngine;
5  using UnityEngine.SceneManagement;
6 #if UNITY_EDITOR
7  using UnityEditor.SceneManagement;
8 #endif
9
10
11 public class GameSystem : MonoBehaviour
12 {
13     public static GameSystem Instance { get; private set; }
14
15     static int s_CurrentEpisode = -1;
16     static int s_CurrentLevel = -1;
17
18     public GameObject[] StartPrefabs;
19     public float TargetMissedPenalty = 1.0f;
20     public AudioSource BGMPPlayer;
21     public AudioClip EndGameSound;
22
23     public float RunTime => m_Timer;
24     public int TargetCount => m_TargetCount;
25     public int DestroyedTarget => m_TargetDestroyed;
26     public int Score => m_Score;
27
28     float m_Timer;
29     bool m_TimerRunning = false;
30
31     int m_TargetCount;
32     int m_TargetDestroyed;
33
34     int m_Score = 0;
35
36     void Awake()
37     {
38         Instance = this;
39         foreach (var prefab in StartPrefabs)
40         {
41             Instantiate(prefab);
42         }
43     }
44 }
```

```

43
44         PoolSystem.Create();
45     }
46
47     void Start()
48     {
49         WorldAudioPool.Init();
50
51         RetrieveTargetsCount();
52
53 #if UNITY_EDITOR
54         //in the editor we find which level we are currently in. Inefficient but
55         //since any level can be opened in the
56         //editor we can't assume where we start
57
58         string currentScene = SceneManager.GetActiveScene().path;
59         for (int i = 0; i < GameDatabase.Instance.episodes.Length &&
60         s_CurrentEpisode < 0; ++i)
61         {
62             for (int j = 0; j < GameDatabase.Instance.episodes[i].scenes.Length;
63             ++j)
64             {
65                 if (GameDatabase.Instance.episodes[i].scenes[j] == currentScene)
66                 {
67                     s_CurrentEpisode = i;
68                     s_CurrentLevel = j;
69                     break;
70                 }
71             }
72         }
73
74 #else
75         //in the final game, we init everything to episode & level 0, as we
76         //can't start from somewhere else
77         if(s_CurrentEpisode < 0 || s_CurrentLevel < 0)
78         {
79             s_CurrentEpisode = 0;
80             s_CurrentLevel = 0;
81         }
82
83     public void ResetTimer()
84     {
85         m_Timer = 0.0f;
86     }
87
88     public void StartTimer()
89     {
90         m_TimerRunning = true;
91     }
92
93     public void StopTimer()
94     {
95         m_TimerRunning = false;
96     }

```

```

97
98     public void FinishRun()
99     {
100         BGMPPlayer.clip = EndGameSound;
101         BGMPPlayer.loop = false;
102         BGMPPlayer.Play();
103
104         Controller.Instance.DisplayCursor(true);
105         Controller.Instance.CanPause = false;
106         FinalScoreUI.Instance.Display();
107     }
108
109     public void NextLevel()
110     {
111 #if UNITY_EDITOR
112         //in editor if we didn't found the current episode or level, mean we are
113         //playing a test scene not part of the
114         //game database list, so calling next level is the same as restarting
115         //level
116         if (s_CurrentEpisode < 0 || s_CurrentLevel < 0)
117         {
118             var asyncOp =
119             EditorSceneManager.LoadSceneAsyncInEditMode(EditorSceneManager.GetActiveScene().path, new LoadSceneParameters(LoadSceneMode.Single));
120             return;
121         }
122 #endif
123
124         s_CurrentLevel += 1;
125
126         if (GameDataInstance.episodes[s_CurrentEpisode].scenes.Length <=
127             s_CurrentLevel)
128         {
129             s_CurrentLevel = 0;
130             s_CurrentEpisode += 1;
131         }
132
133         if (s_CurrentEpisode >= GameDatabase.Instance.episodes.Length)
134             s_CurrentEpisode = 0;
135
136 #if UNITY_EDITOR
137         var op =
138         EditorSceneManager.LoadSceneAsyncInEditMode(GameDatabase.Instance.episodes[s_CurrentEpisode].scenes[s_CurrentLevel], new
139         LoadSceneParameters(LoadSceneMode.Single));
140     #else
141
142         SceneManager.LoadScene(GameDatabase.Instance.episodes[s_CurrentEpisode].scenes[s_CurrentLevel]);
143     #endif
144     }
145
146     void RetrieveTargetsCount()
147     {
148         var targets = Resources.FindObjectsOfTypeAll<Target>();
149
150         int count = 0;

```

```

145
146         //The spawner create their target and disable them before that function
147         //run, so retrieving all Target will also
148         //retrieve the one the spawner will use.
149         foreach (var t in targets)
150     {
151 #if UNITY_EDITOR
152
153         //in editor we have to check if the target returned is not a prefab,
154         //since Resouces.FindObjectofTypeAll return
155         //also loaded prefab. In the player no need as there is no prefab
156         //loaded.
157
158         //if the scene isn't valid it's a prefab.
159         if (!t.gameObject.scene.IsValid())
160     {
161         continue;
162     }
163 #endif
164
165         //we only count target with positive point, as negative point you
166         //have to avoid destroying them
167         if (t.pointValue > 0)
168             count += 1;
169
170     }
171
172     m_TargetCount = count;
173     m_TargetDestroyed = 0;
174     m_Score = 0;
175
176     GameSystemInfo.Instance.UpdateScore(0);
177     LevelSelectionUI.Instance.Init();
178 }
179
180 void Update()
181 {
182     if (m_TimerRunning)
183     {
184         m_Timer += Time.deltaTime;
185
186         GameSystemInfo.Instance.UpdateTimer(m_Timer);
187     }
188
189     Transform playerTransform = Controller.Instance.transform;
190
191     //UI Update
192     MinimapUI.Instance.UpdateForPlayerTransform(playerTransform);
193
194     if(FullscreenMap.Instance.gameObject.activeSelf)
195         FullscreenMap.Instance.UpdateForPlayerTransform(playerTransform);
196 }
197
198 public float GetFinalTime()
199 {
200     int missedTarget = m_TargetCount - m_TargetDestroyed;

```

```
199         float penalty = missedTarget * TargetMissedPenalty;
200
201         return m_Timer + penalty;
202     }
203
204
205     public void TargetDestroyed(int score)
206     {
207         m_TargetDestroyed += 1;
208         m_Score += score;
209
210         GameSystemInfo.Instance.UpdateScore(m_Score);
211     }
212 }
```

7.9 游戏结束判断

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class EndCheckpoint : MonoBehaviour
6  {
7      void OnTriggerEnter(Collider other)
8      {
9          GameSystem.Instance.StopTimer();
10         GameSystem.Instance.FinishRun();
11         Destroy(gameObject);
12     }
13 }
14
```