基于unity制作的3D大富翁游戏（Exploit）设计

目录

[基于unity制作的3D大富翁游戏（Exploit）设计 1](#_Toc137688144)

[摘要 2](#_Toc137688145)

[一、课题研究意义： 3](#_Toc137688146)

[1.1 基于unity开发的大富翁游戏的市场价值分析： 3](#_Toc137688147)

[二、用户的需求分析： 3](#_Toc137688148)

[三、游戏内容设计 4](#_Toc137688149)

[3.1 人物脚本 4](#_Toc137688150)

[3.2金钱的实时显示 6](#_Toc137688151)

[3.3游戏内地皮交易 7](#_Toc137688152)

[3.4游戏内时间系统设计 10](#_Toc137688153)

[3.5游戏内玩家胜利系统设计 10](#_Toc137688154)

[四、人物移动设计 11](#_Toc137688155)

[4.1地图方块设计 11](#_Toc137688156)

[4.2角色移动设计 11](#_Toc137688157)

[4.3玩家行为设计 12](#_Toc137688158)

[4.4骰子设计 12](#_Toc137688159)

[五、地图和界面设计 13](#_Toc137688160)

[5.1地形设计 13](#_Toc137688161)

[5.2 设计界面： 14](#_Toc137688162)

[开始界面： 14](#_Toc137688163)

[菜单页面： 16](#_Toc137688164)

[购买和售出页面： 16](#_Toc137688165)

[六、摄像机跟随 17](#_Toc137688166)

# 摘要

大富翁是一款家喻户晓的桌面游戏，最初由美国一家玩具公司于1935年发明并推出。随着时间的推移，大富翁逐渐成为世界各地人们喜爱的经典游戏之一。现代社会中，人们对游戏的需求越来越高，他们希望游戏能够提供趣味性、挑战性和社交性等多种元素，同时也期望游戏能够在视觉效果和游戏体验方面不断提升。

Unity是一款流行的跨平台游戏引擎，具有强大的可视化编辑器和可定制化的工作流程。Unity不仅适用于移动游戏、PC游戏和主机游戏等各种游戏类型，而且还可以为虚拟现实和增强现实应用程序提供支持。Unity平台的优势包括易于上手、高度灵活、快速迭代和强大的图形渲染引擎等方面。

基于Unity引擎开发的大富翁游戏可以为玩家带来更加真实、丰富的游戏体验，包括3D图形、物理模拟和实时网络多人游戏等。

总之，本小组基于Unity开发的大富翁游戏具有无限的创新空间和丰富的游戏体验，它将为玩家带来全新的游戏乐趣和社交互动体验，是一款备受期待的现代游戏。

关键词：Unity

# 一、课题研究意义：

## 1.1 基于unity开发的大富翁游戏的市场价值分析：

1.市场需求： 大富翁是一款经典的家庭游戏，它有着广泛的受众群体。大富翁游戏不仅适合家庭娱乐，也适合团队建设和教育。由于移动设备和PC游戏市场的不断扩大，大富翁游戏在这些平台上的开发和发行有着很大的市场需求。

2.竞争情况： 大富翁是一款经典游戏，市场上已经存在许多不同版本的大富翁游戏。因此，在开发大富翁游戏时需要考虑如何突出差异化和创新，以吸引更多的用户。

3.技术要求： 大富翁游戏涉及到的技术要求并不是非常高。Unity引擎提供了丰富的功能和工具，使得开发大富翁游戏变得相对简单。开发者只需要具备一定的编程能力和游戏设计经验即可。

4.市场潜力： 大富翁游戏在市场上有着很大的潜力。如果能够开发出一款有创意、有趣、易于上手的大富翁游戏，并且在市场上做好推广和宣传，就有可能取得不错的市场表现。

综上所述，基于Unity开发的大富翁游戏在市场上有一定的价值，但需要开发者在市场竞争中突出差异化和创新，以吸引更多的用户。

# 二、用户的需求分析：

本游戏为单机游戏，其面对的用户群体为热衷于单机游戏的广大pc端群体，这个群体的年龄大小不一，性别男女都有。针对对大富翁游戏的抽样调查并结合我们自己对pc端游戏的个人理解，我们小组总结出如下功能：

1玩家形象

玩家在游戏开始前选择要使用的替身形象

2游戏页面

我们设计了游戏开始和结束页面，交易界面，菜单界面并实现了相应功能

3金币系统

玩家在游戏中可以通过随机事件、房屋的买卖等方式获取和消耗虚拟的金币

4随机方块

我们在地图上设计了宝箱，机会，转盘等随机方块，玩家走到上面可以随机获得或扣除金币

5行动

游戏中玩家按回合制行动，每个回合每个玩家行动一次，不能放弃行动。按骰子的点数在地图上移动，移动到目的地后购买土地、支付费用或触发随机事件

6土地系统

玩家行动过后，若所处的方块上的土地没有所有者，玩家可以购买土地，购买后的土地会变成代表购买者相应的颜色。若方块上的土地有所有者，则玩家需要支付相应的金币

8游戏结算

当玩家的金币小于等于0时，则会被判出局；金币大于某个数时，则被判获胜

# 三、游戏内容设计

我把大部份的变量都设置成了public，主要是本人对public和private的适用范围不太清晰，为了以防出错，我把所有public声明的变量名字都不一样

## 3.1 人物脚本

下图是人物挂载的脚本



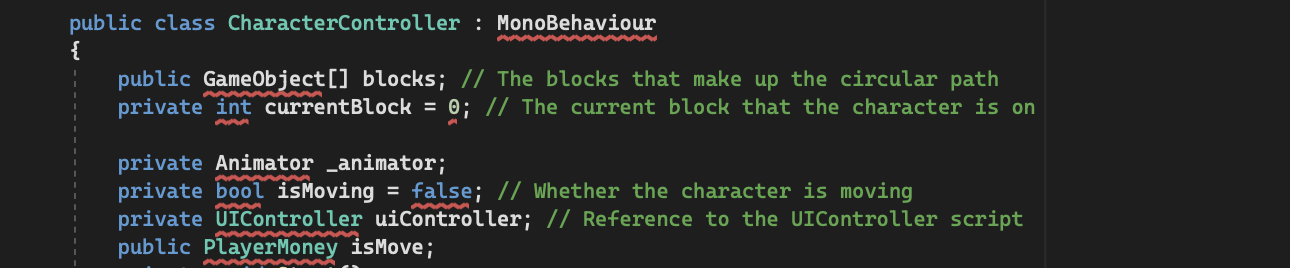
第一个脚本Character Controller在第六节人物移动设计中展示

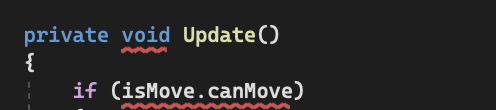


Play Money脚本如下

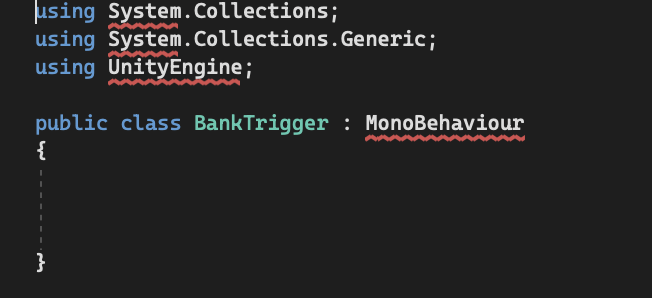


主要实现的功能就是记录人物的金钱以及人物经过带有‘bank’标签的起点时获得金钱，以及到判断到一定数额金钱后弹出获胜界面并且使角色停止移动。我在CharacterController中加入的这个判断条件，使之成为所有人物行动的前提这段代码可能不会在第六节中提到。





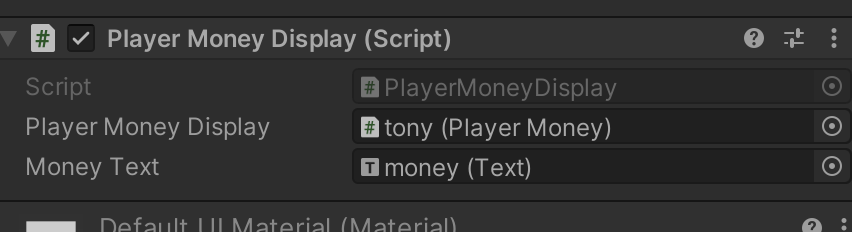
引用的bank Trigger代码如下，是一个空脚本。

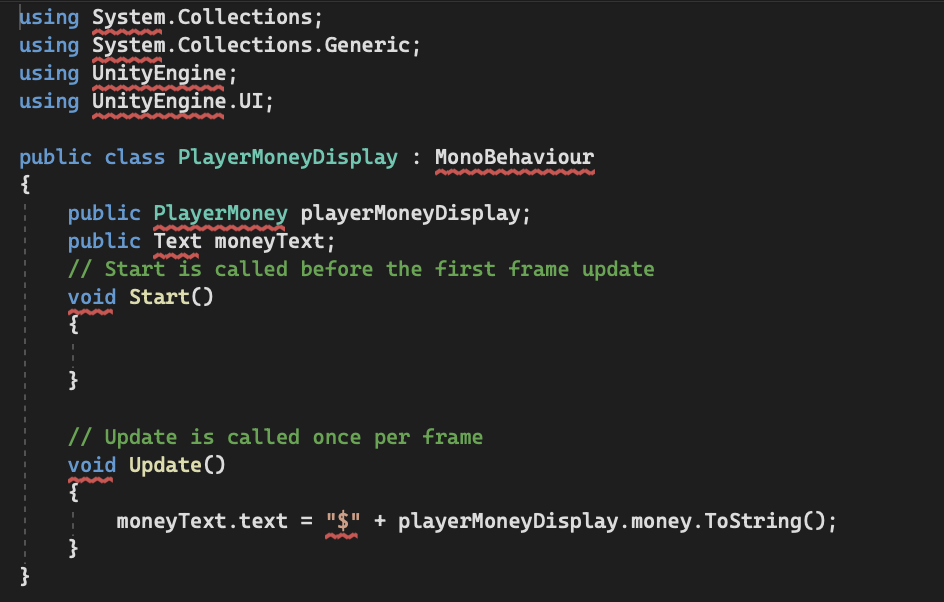


GetPlayer Position（）主要是获取人物位置，这个方法主要为了实现角色对于地皮的交易

## 3.2金钱的实时显示

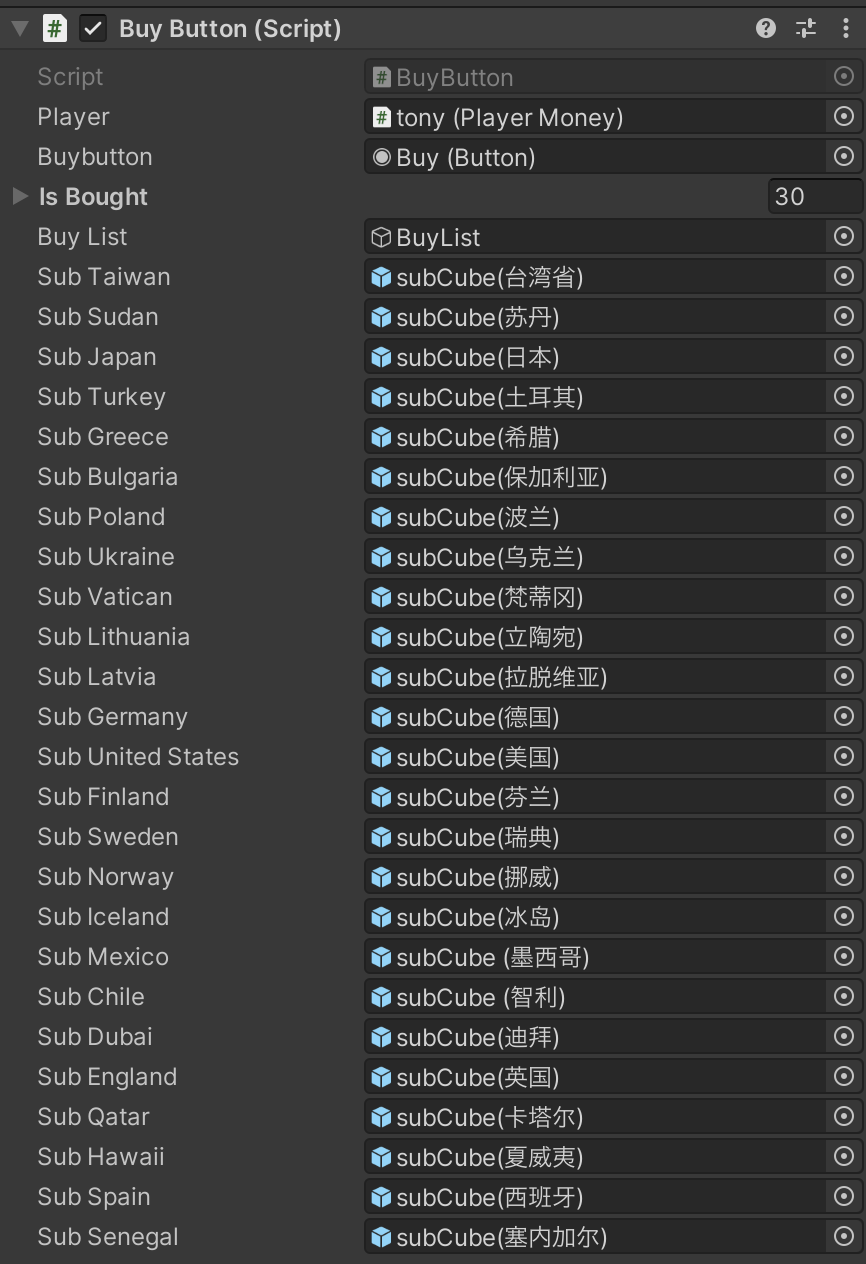
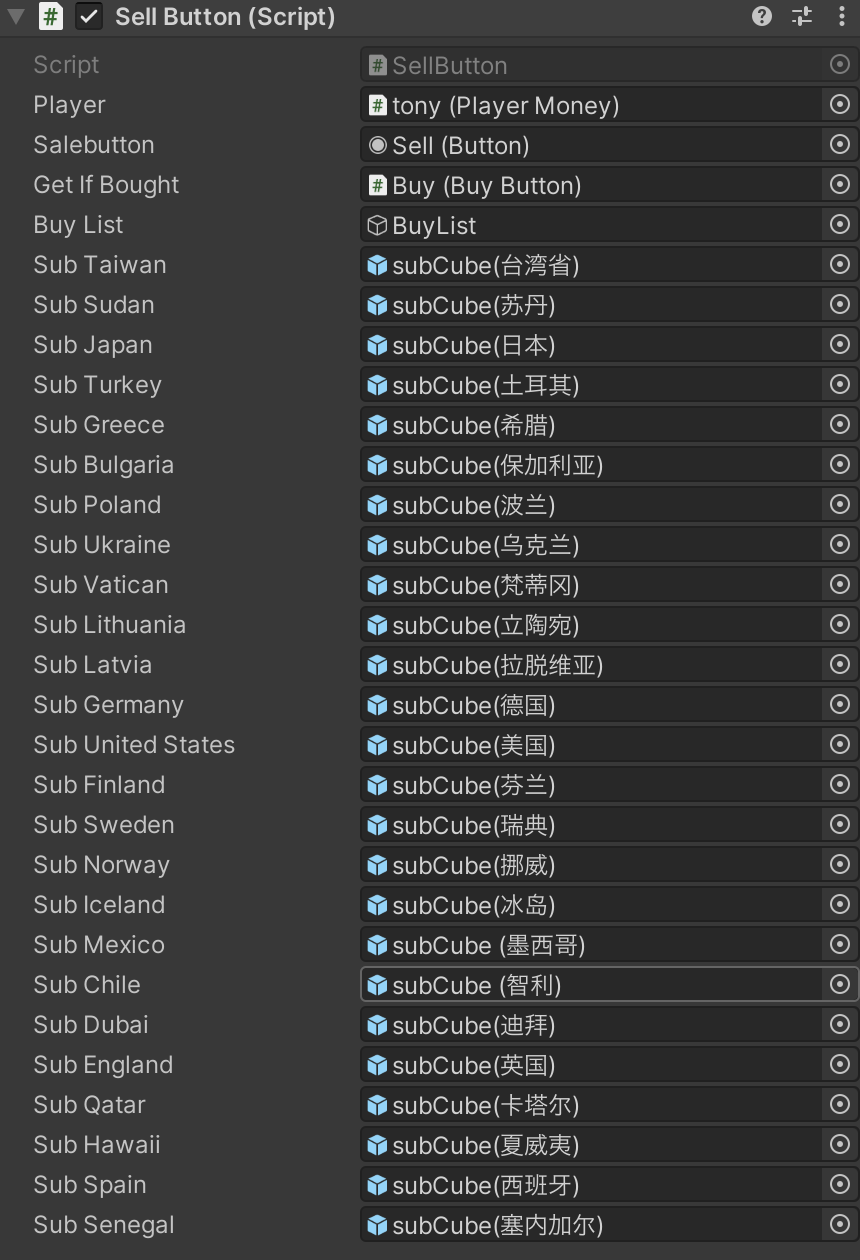
通过引用Player money脚本的属性来实现，并把这个脚本挂载在text UI上，代码和text UI如下图所示



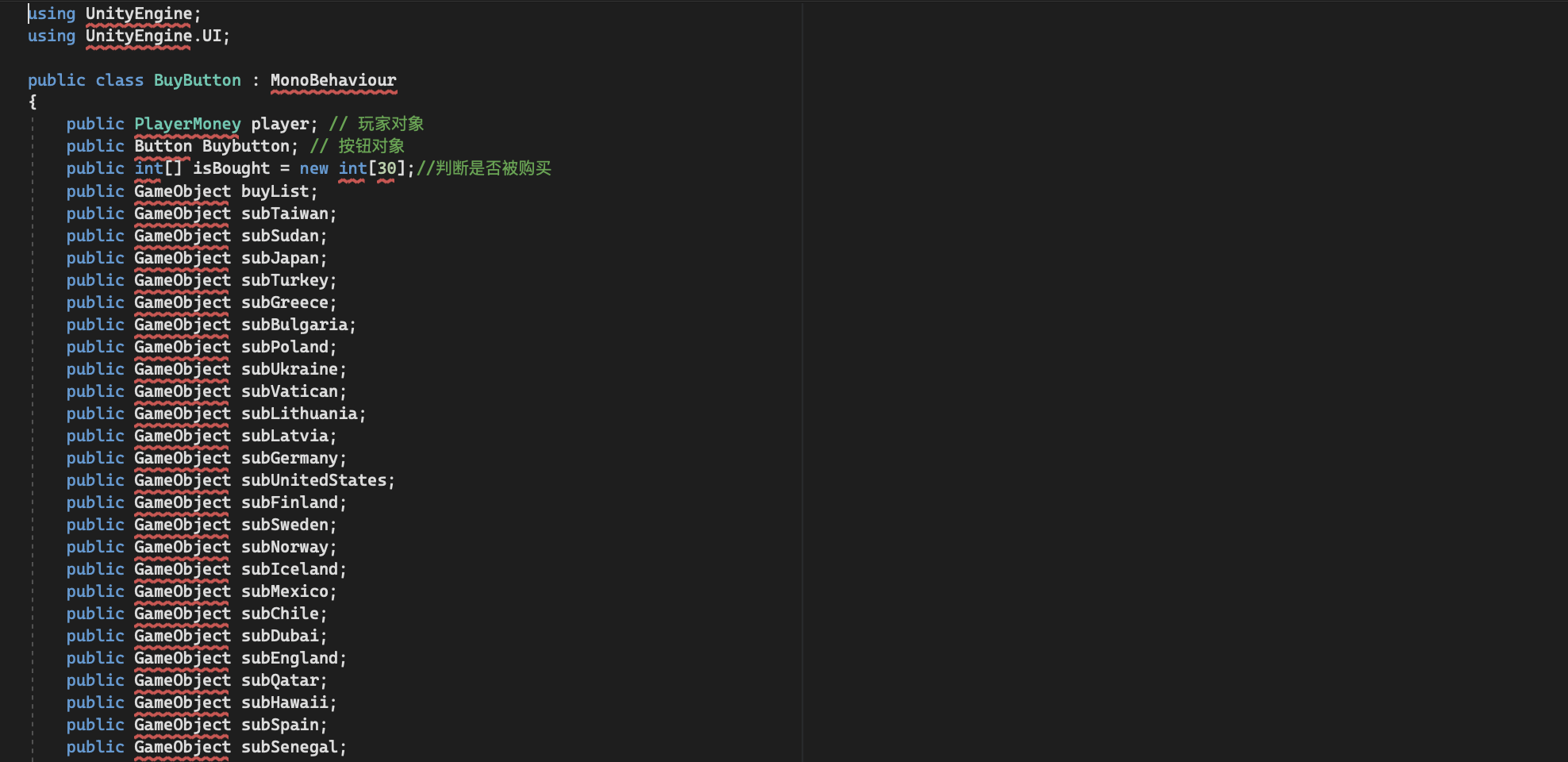


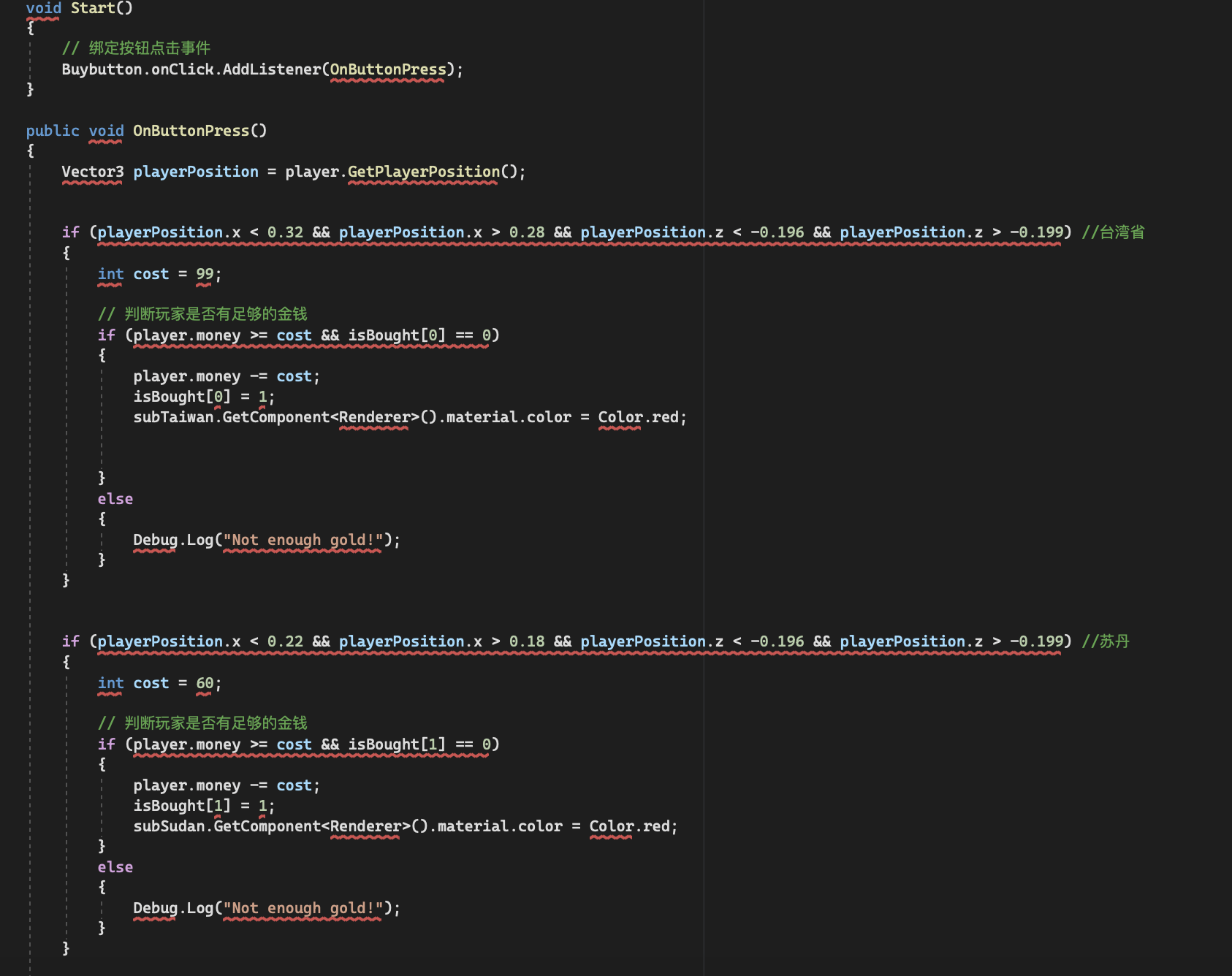
## 3.3游戏内地皮交易

我们使用了点击buy按钮来进行对每一块地皮的购买和sale按钮来实现对每一块地皮的售出。我们通过人物目前所在的坐标来判断人物在哪块地皮上。以及通过数组来记录这块地皮是否被购买。以及我们实现了点击购买之后地皮的颜色变换。并实现了点击buy和sale按钮后界面自动退出的效果。下图展示了脚本挂载在购买/售出列表的效果。

下图是BuyButton的具体代码



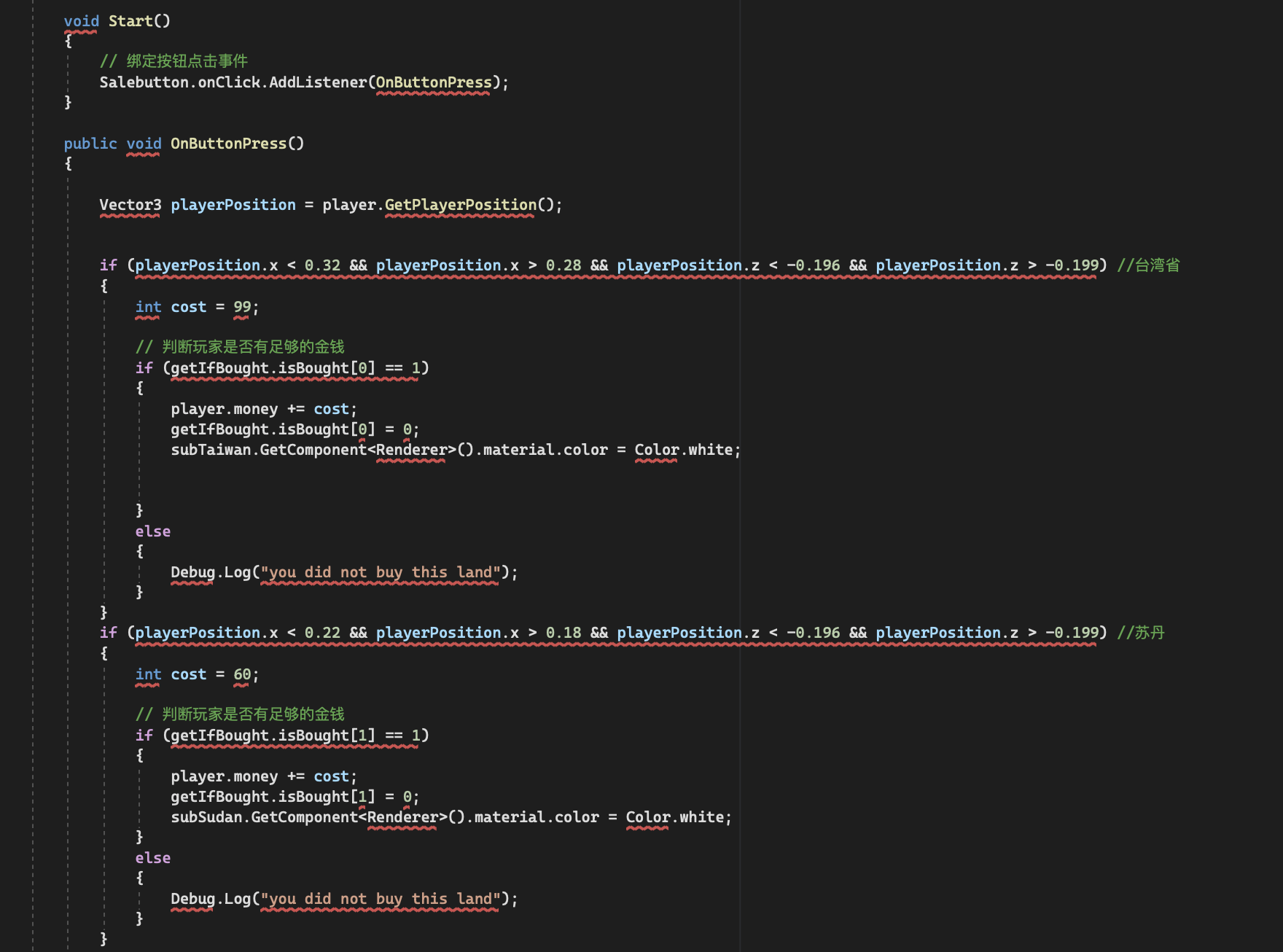




上图的代码并不完整，但是这些代码能够体现出我上文所写的实现购买的核心思想。

出售的代码则是引用了购买代码的is bought变量来判断是否被购买。下图则是sellbutton的代码（不完整）。



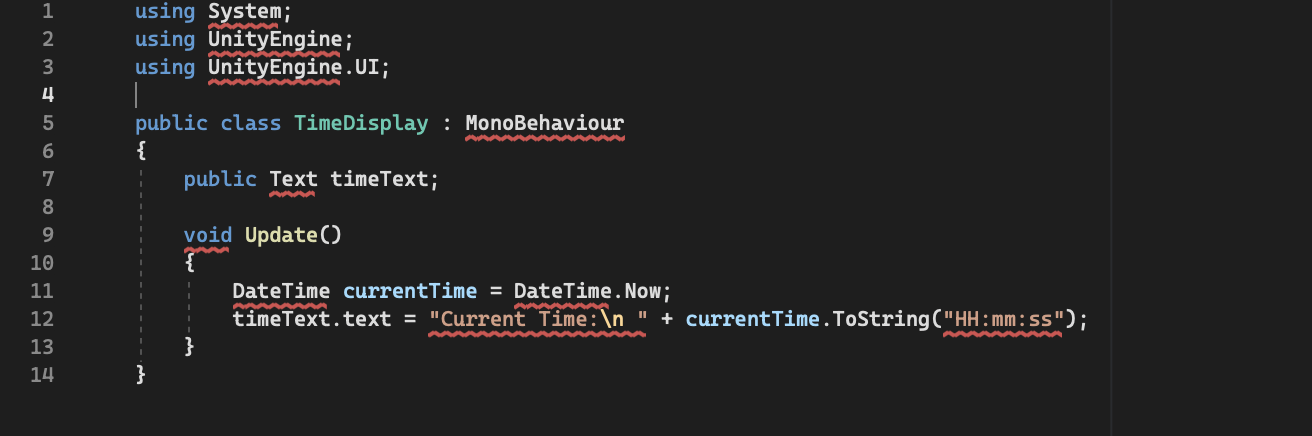




自此实现了土地的购买与售出。如过后续有多人玩家，则可以把is bought变量设置成其他数字。

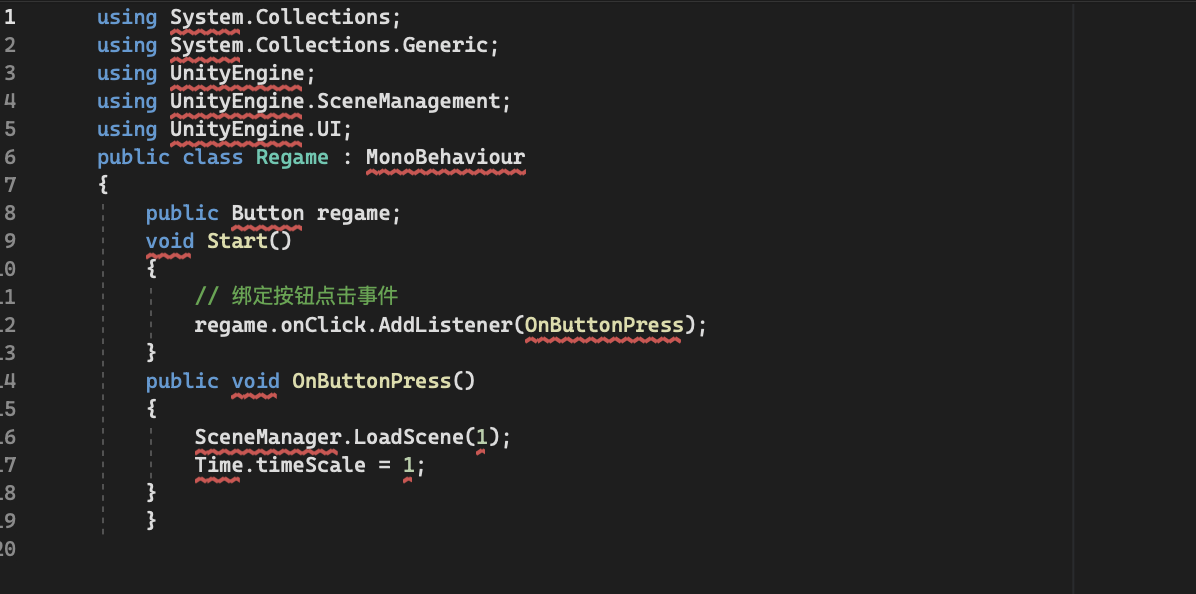
## 3.4游戏内时间系统设计

引用了unity自带的date Time来获取实时时间，并通过Time Display脚本来显示到UI界面上



## 3.5游戏内玩家胜利系统设计

当人物金钱达到某一值时，则会弹出获胜界面。具体脚本在上图的PlayerMoney脚本中体现。当玩家获胜后，可以regame来实现重新开始游戏。脚本和挂载效果如下图

# 四、人物移动设计

## 4.1地图方块设计

地图由36个相互分离的方块组件构成，在控制玩家移动行为的CharacterController脚本（挂载在玩家控制的角色Tony身上）中创建了一个名为Block的数组，其中存储了36个地块对应的Cube组件。

## 4.2角色移动设计

玩家的角色行为由挂载在玩家控制的角色（Tony）身上的脚本文件“CharacterController”控制和决定，在此脚本中运用到了以下方法和元素：

1. **blocks**（GameObject数组）：表示构成环形路径的方块对象。
2. **currentBlock**（整数）：记录角色当前所在的方块索引。
3. **\_animator**（Animator）：用于控制角色的动画组件。
4. **isMoving**（布尔值）：标识角色是否正在移动。
5. **uiController**（UIController）：用于管理UI界面的控制器。

在**Start()**方法中：

* **\_animator**获取挂载脚本的游戏对象上的Animator组件。
* **uiController**通过FindObjectOfType方法在场景中查找并获取UIController脚本实例。

在**Update()**方法中：

* 如果角色正在移动，则直接返回，不处理玩家的输入。
* 如果按下空格键（KeyCode.Space）：
  + **diceNumber**通过调用Random.Range方法生成一个1到6的随机数，模拟掷骰子的结果。
  + **uiController.ShowDiceNumber(diceNumber)**用于在UI界面上显示骰子的点数。
  + 调用**StartCoroutine(MoveToNextBlock(diceNumber))**，启动协程，将角色移动到下一个方块。
* 如果按下回车键（KeyCode.Return）并且角色没有在移动中，则执行以下操作：
  + **uiController.ToggleBuyMenu()**用于在UI界面上切换购买菜单的显示状态。

**MoveToNextBlock(int steps)**是一个协程方法，用于将角色移动到下一个方块。它包含以下步骤：

* 将**isMoving**标志设置为**true**，表示角色正在移动。
* 将动画参数**isRunning**设置为**true**，播放奔跑动画。
* 通过循环执行以下操作，使角色逐步移动到下一个方块：
  + **targetPosition**获取下一个方块的位置。
  + 在角色和目标位置之间，通过循环不断移动角色的位置，直到两者之间的距离小于0.01f。
  + 更新角色的朝向，使其面向移动方向。
  + 当角色到达下一个方块时，更新**currentBlock**为下一个方块的索引。
* 将动画参数**isRunning**设置为**false**，停止奔跑动画。
* 将**isMoving**标志设置为**false**，表示角色移动结束。

## 4.3玩家行为设计

在CharacterController脚本中，声明了用以控制玩家行为的动画器（Animator），用以控制玩家在移动时的行为表现。  
在Unity应用程序中，我们可以通过“动画器”窗口制定玩家控制的角色的动作逻辑，如图：  
图片包含 日程表

描述已自动生成

当玩家因为骰子点数的改变而需要移动到下一个指定的方块使，动画器即会进行由idle（静止状态），向run（奔跑状态）的过渡。我们取消了过渡时间，以便在游戏过程中能实现tony能更流畅地向前奔跑。

## 4.4骰子设计

采用Rigidbody组件Velocity控制移动 骰子需要挂载Rigidbody组件

设计要点：

public float throwForce = 1f;：用于设置扔骰子时施加的力度。

public UIController uiController;：用于引用UI控制器，可以调用其中的方法来显示和隐藏UI元素。

public int diceNumber;：用于存储骰子的点数。

private Rigidbody \_rigidbody;：用于获取骰子的刚体组件，以便对其施加力和进行碰撞检测。

Start() 方法：

在Start()方法中，通过获取组件引用来初始化\_rigidbody变量。

Update() 方法：

在Update()方法中，通过按下空格键触发骰子扔出的操作。

如果按下了空格键，则调用ThrowDice()方法。

ThrowDice() 方法：

ThrowDice()方法用于扔骰子并生成随机点数。

首先，将骰子的位置和旋转重置为初始状态。

然后，给骰子施加一个随机的力，使其向上扔出。这里使用AddForce()方法，并设置为冲量模式（ForceMode.Impulse）。

接下来，设置骰子的碰撞检测模式为Continuous，以便能够检测到与平面的碰撞。

生成随机点数，并调用uiController的ShowDiceNumber()方法来显示点数。

OnCollisionEnter() 方法：

OnCollisionEnter() 方法用于处理骰子与其他物体的碰撞事件。

如果骰子与标签为Chessboard的物体发生碰撞，表示骰子碰撞到了平面。在这种情况下，通过使用Reflect()方法，根据碰撞点的法线来实现骰子的弹跳效果。

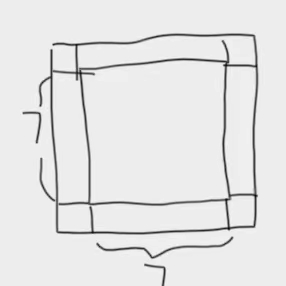
最后，根据您的需求和游戏中的具体情况，将脚本挂载在骰子对象上，并将所需的UI控制器分配给uiController变量。请确保在场景中存在名为Ground的平面对象，以便骰子在碰撞时能够触发弹跳效果。

# 五、地图和界面设计

## 5.1地形设计

1、设计地图

首先我们规划了地图的大小为9x9（后来改成10x10）



然后按照预期搭建了地图



然后在其中加入了画布和UI控件来使其美化

最终通过不断完善和美化达到我们的预期。

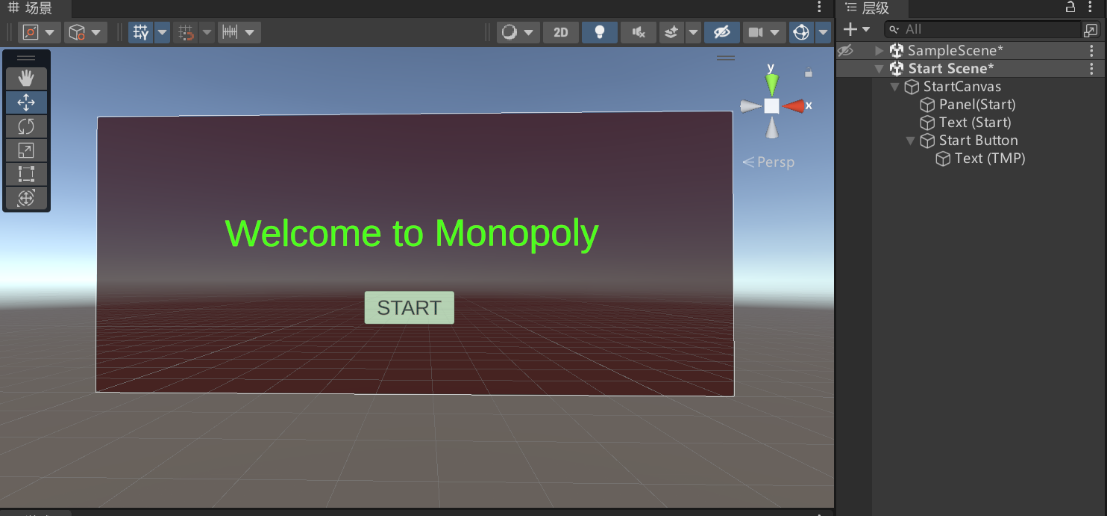
## 5.2 设计界面：

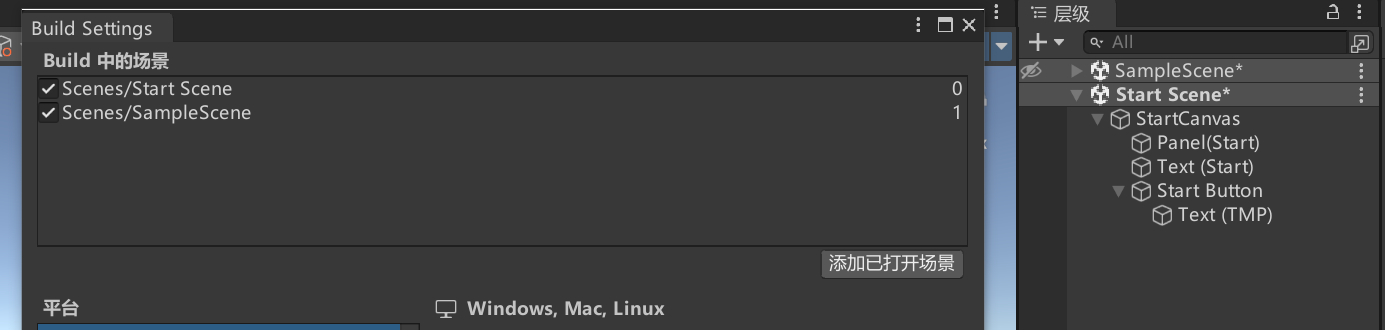
### 开始界面：

地图

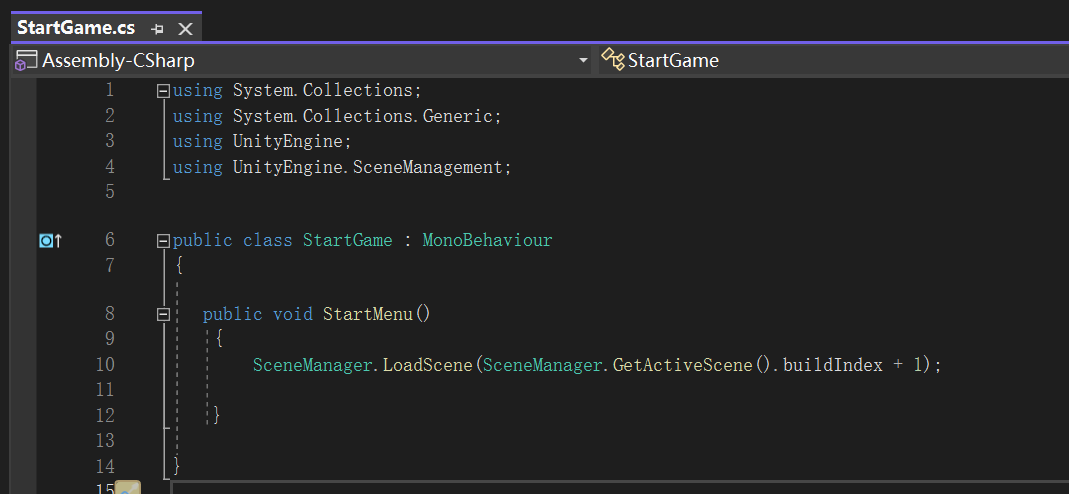
描述已自动生成

我创建了一个新的Scene来制作开始页面。



然后我们在生成设置里面将Start Scene设为0，将Sample Scene设为1。这里是运行后的打开顺序，0是最先打开的场景。

然后开始带入我们写的脚本。



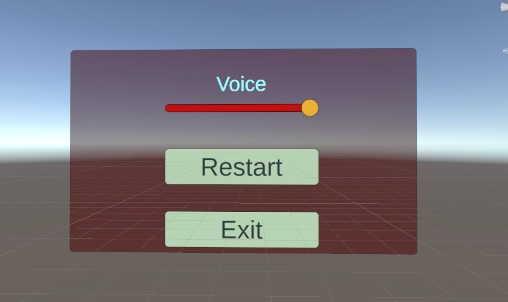
将脚本加在开始页面的Start按钮上



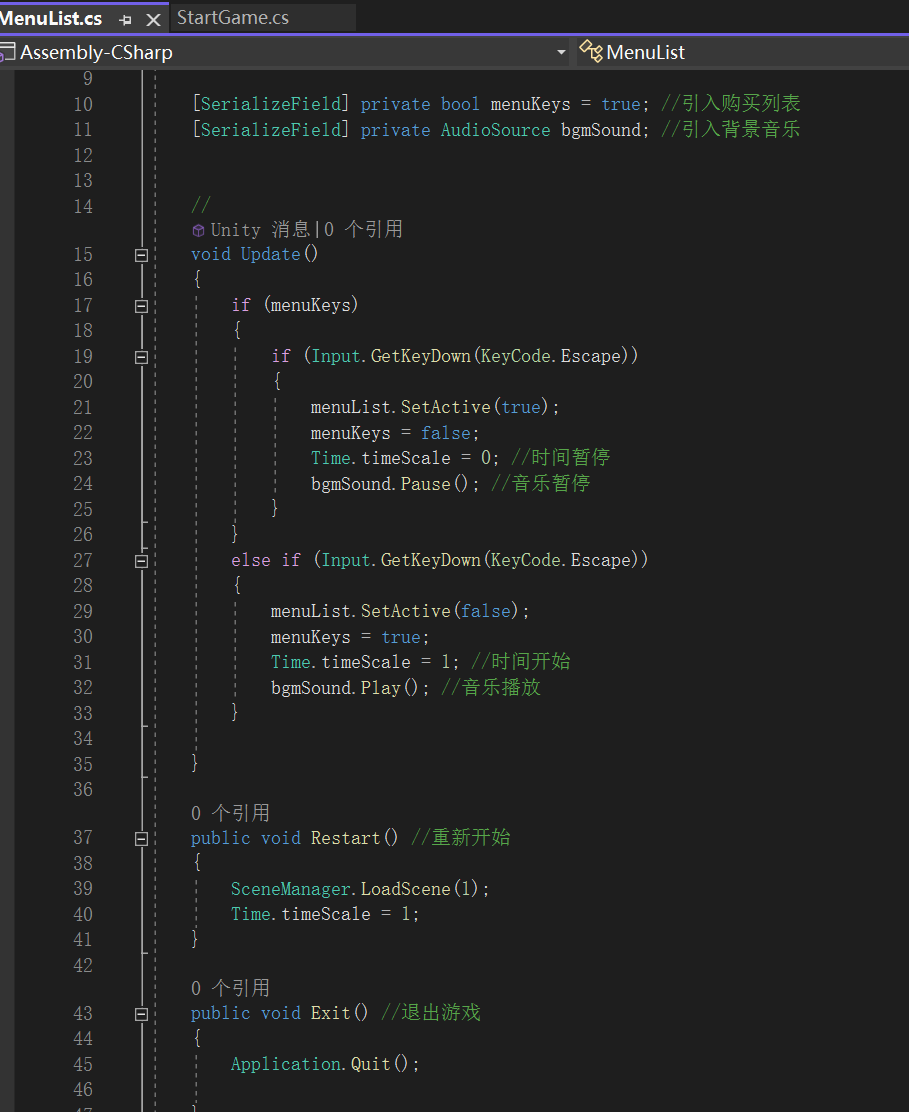
脚本中SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);意味点击按钮后场景切换为原场景编号+1，也就是从Start Scene 切换到 Sample Scene，实现场景转换（结束页面同理）。

### 菜单页面：

在Sample Scene新创建一个UI Canvas，在其中添加一个pane来进行设计。

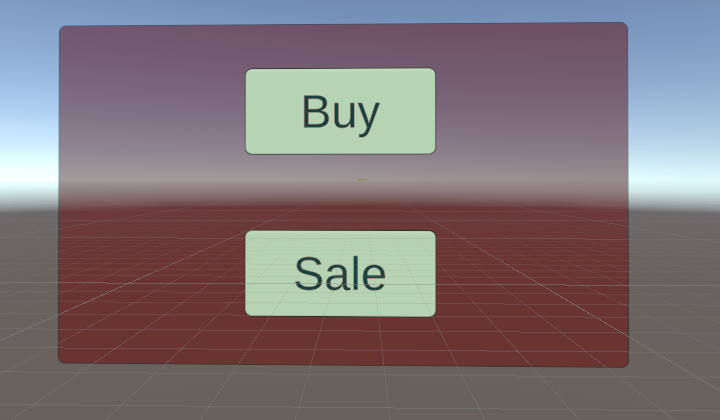


接下来写脚本并带入实现按钮功能，并且实现可以通过按ESC键来唤醒或关闭菜单界面，并且添加了当菜单页面打开时，游戏暂停的功能。



### 购买和售出页面：

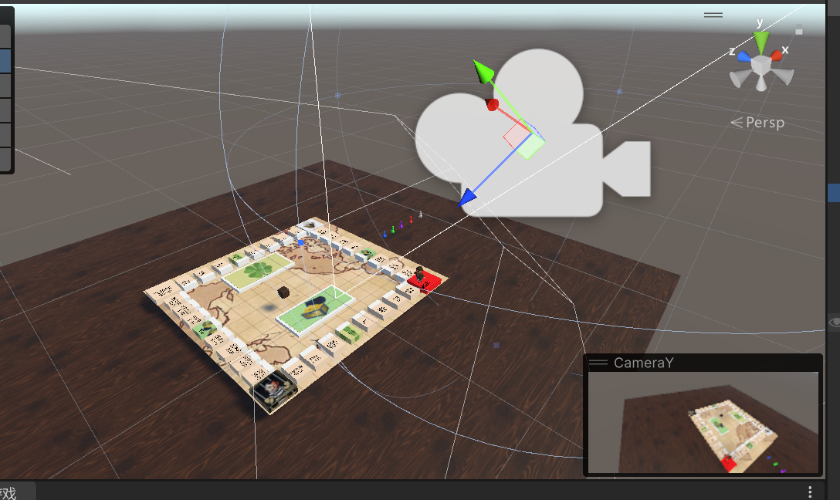
购买和出售页面设计和上面同理。



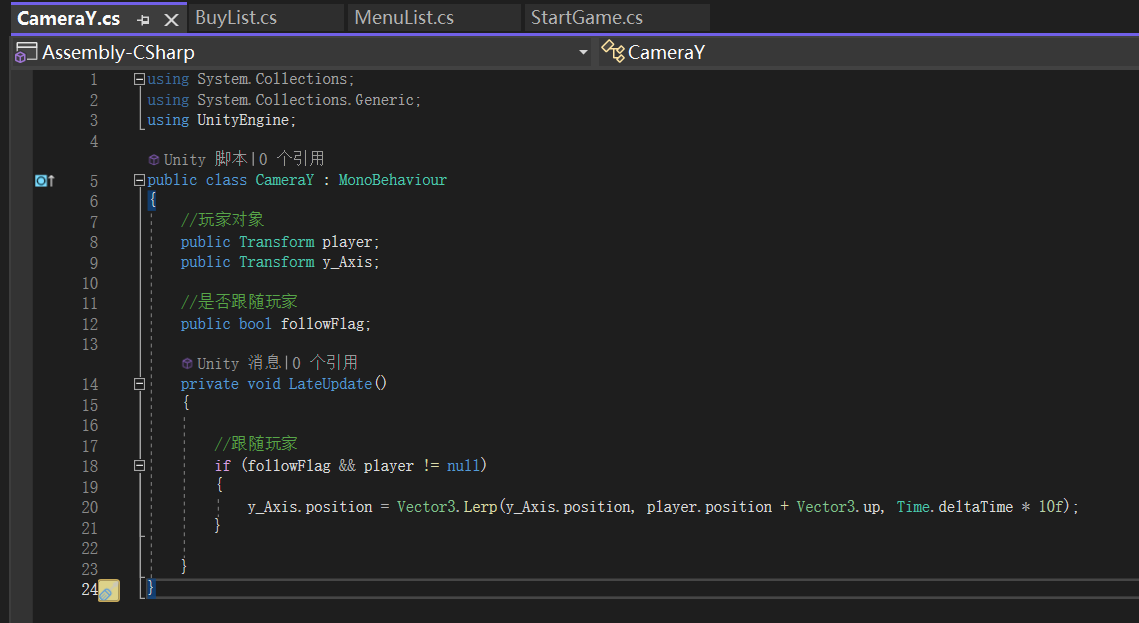
# 六、摄像机跟随

为了更方便玩家观察，我将摄像机设计了可以跟随玩家移动的功能。

首先需要将摄像机的位置调到玩家人物的斜后上方较好观察的视野，如图。



接下来开始编写脚本。



然后将脚本带入摄像机中，从而实现镜头跟随的功能。