技术大作业-TNT卡丁车实验报告

李晓文 2020214437

摘要信息

本作业基于卡丁车demo实现了一个多人联机卡丁车游戏。游戏的核心卖点支持多人比赛（使用桢同步广播的方式，因此可以支持1-100人玩），并且不需要玩家（也就是老师）搭建服务器，因为服务器部署于腾讯云上因此只要有因特网的地方就可以进行联网游戏。

本游戏分为两种模式，分别为竞速模式和积分模式。竞速模式比的是谁先到达终点，而积分模式则是比谁的积分更高。游戏中可以通过加速垫，金币来获取分数，当然如果遇到减速垫则会降低分数。

此外本游戏将原有的基于wasd操作的卡丁车移植到移动端（提供了摇杆的交互形式），并且实现了多种元素的HUD显示，该游戏可以分别储存历史最好的十次成绩（有三种类别，积分，时间和金币数），并可通过UI查看。

本游戏搭建了两个关卡，分为自然世界和古代世界，其中每个场景都具有岩石，植被以及建筑等丰富元素。

老师可以通过在建立房间时选择一个人从而进行单人游戏，不过最好还是使用两台设备进行联机游戏，因为本游戏的开发成本大部分都消耗在联网上了。多人联网是本游戏的核心。

注：创建房间时可以进行人数设置，单人游戏，请务必设置为1.

实现的要素列表

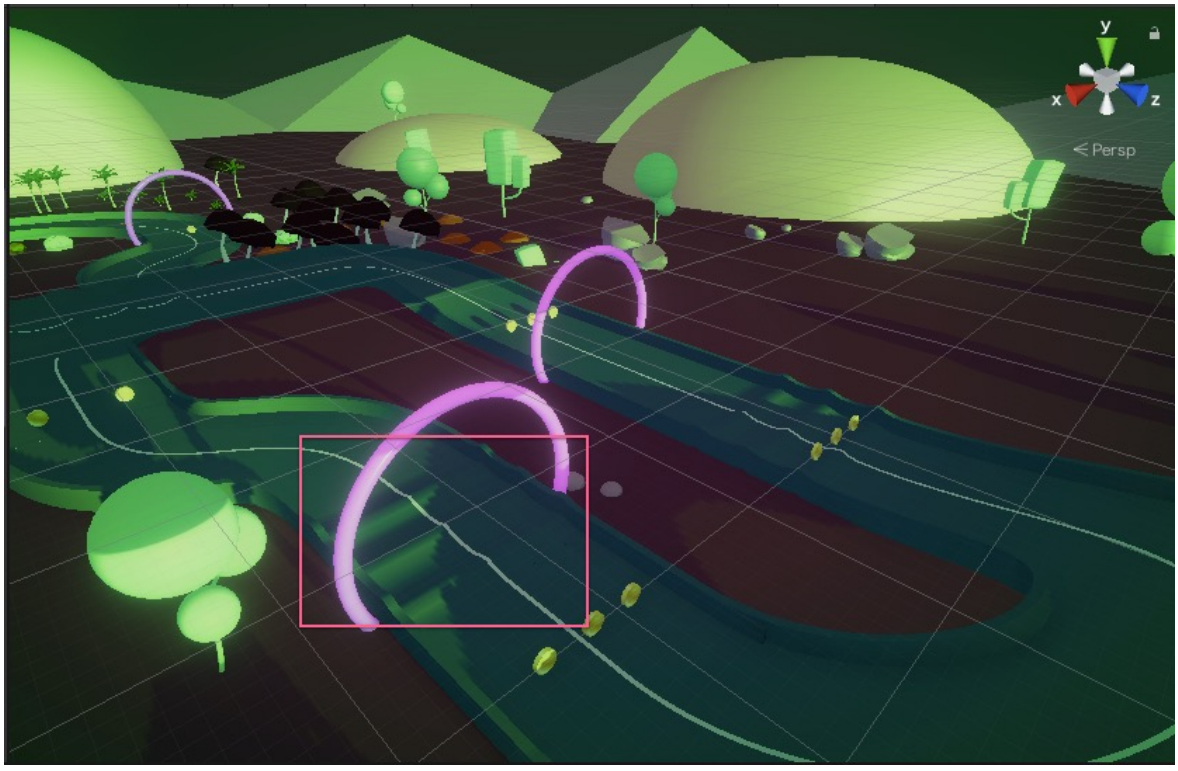
- 改变赛车或者赛道形象（**已实现高分要求**）
- 增加动画，特效（**已实现基本要求**）
- 增加机关 （**已实现高分要求**）
- 保存用户记录 （**已实现高分要求**）
- 音效设计：（**已实现基本要求**）
- 完善UI：（**已实现高分要求**）
- 增加车辆的效果 （**已实现高分要求**）
- 改变游戏的图像效果（**已实现基本要求**）
- 增加获取金币（**已实现高分要求**）
- 双人联机（**难度系数X5**）（**已实现高分要求**）
- 其他加分：**赛道计分板显示当前用的时间分数，以及最高分数**等，时间显示会变化。玩法在一开始就可以进行**模式选择**等

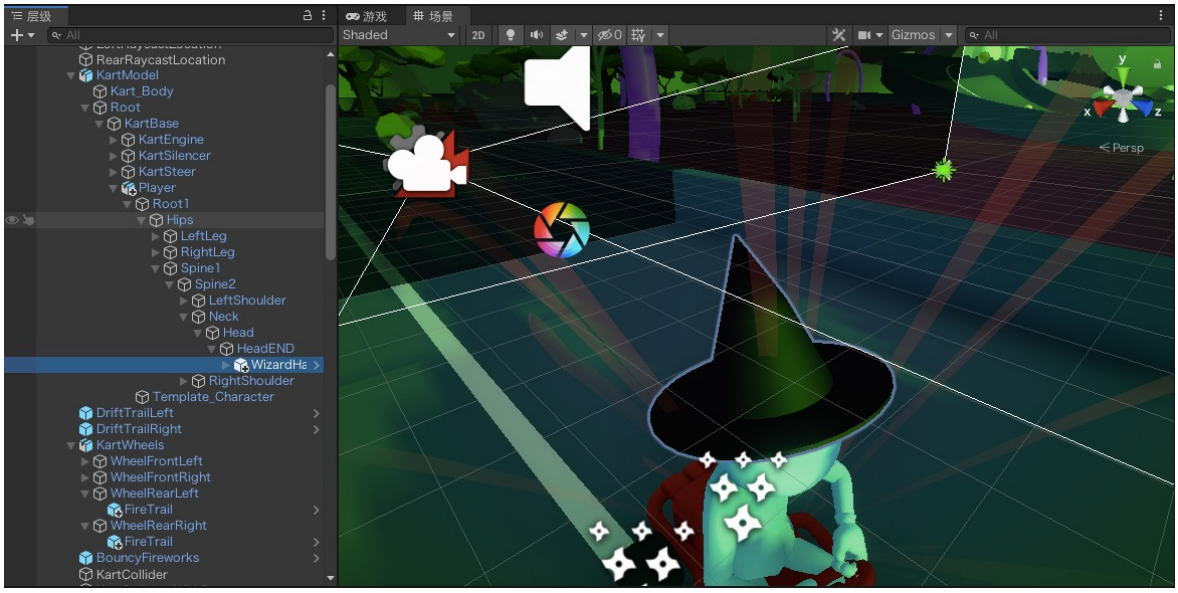
详细描述

- 改变赛车或者赛道形象（**已实现高分要求**）：（**基本要求**）增加了角色挂件，让角色或赛车更酷。（**高分要求**）：赛场布景更加漂亮，分为两个主题，一个是自然世界，一个是古代世界，每个世界对应着一个赛道，并且新赛道上设置了崎岖路面、金币等。

  视频时间：关卡自然世界通关视频.mp4 视频 1min20s

  相关截图：



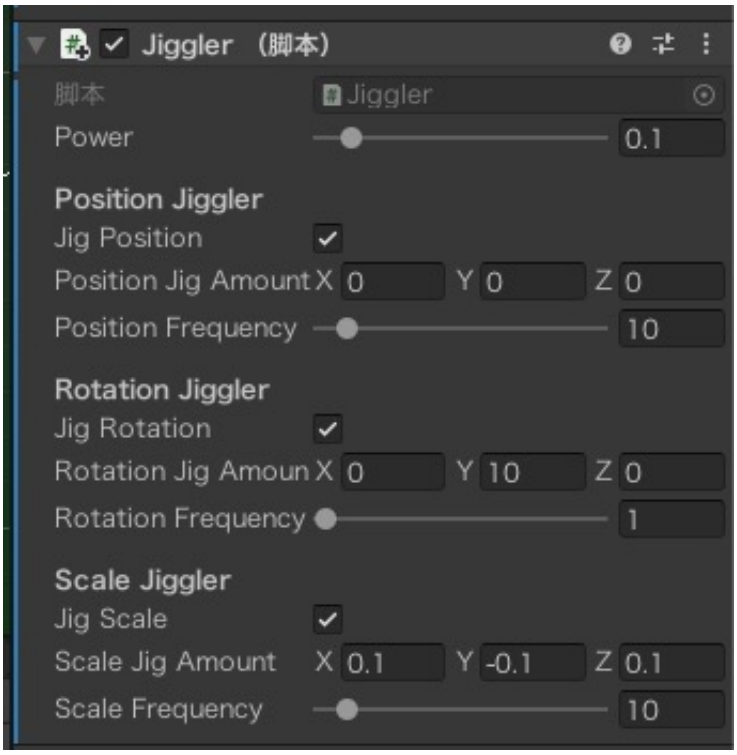


  设计与实现：根据教程，拖拽相关组件即可实现角色挂件功能。金币具体设计实现请看下面的金币实现细节。

- 增加动画，特效（**已实现基本要求**）：实现了树和云的动画。

  视频时间：关卡自然世界通关视频.mp4 视频第40s以后都有

  相关截图：

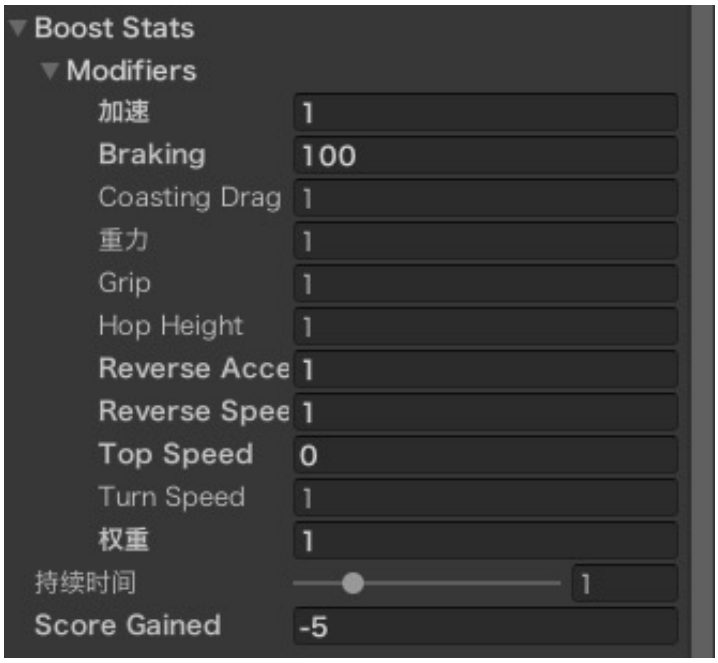


设计与实现：根据官方教程，使用Jiggler实现。

- 增加机关 （**已实现高分要求**）：**（基本要求）**加速垫，上加速。**（高分要求）**除了加速垫，还增加了减速垫，并且加速垫可以增加玩家得分，减速垫除了减速还会减少玩家得分。

  视频时间：关卡自然世界通关视频.mp4 视频1min46s

  相关截图：

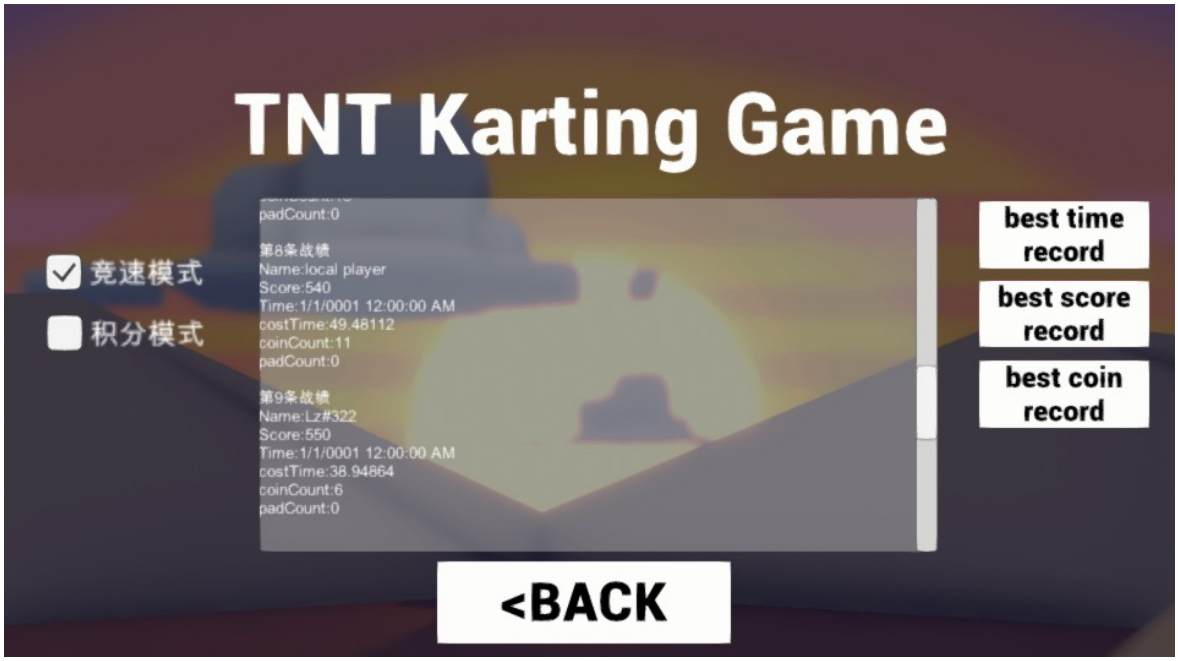


设计与实现：加速垫，根据教程拖拽到场景就可。减速垫则主要修改了braking参数，并且我还增加了分数的影响功能，具体实现可见分数模式。

- 保存用户记录 （**已实现高分要求**）：**（基本要求）**记录用户历史成绩记录。展现用户多次游戏的成绩排行，重新登录后仍然记录有旧的成绩记录。**（高分要求）**：能分别保存最佳分数，最佳时间，最多金币等三类成绩。并且UI上能告知用户成绩排行，创建记录的时间。当保存的记录数超过10条以后，删除掉最后一名的成绩。

  视频时间：功能演示以及关卡古代世界过关失败视频.mp4 视频开始到20s

相关截图：



设计与实现：本游戏会将游戏记录分别以json的文件格式储存到三个文件里面，对应三种不同的记录形式（时间，金币，分数）,记录文件如下图所示：

功能的实现主要在TrackManager和TrackRecord两个类中实现。具体的记录储存和读取代码如下：

```
[Serializable]
    public class BestTimeRecord
```

```
    {
        public List<TrackRecord> records;
        public static SortedList Sortedrecord = new SortedList(new FloatSort());
        public static int BestTimeRecordSize = 10;


        const string k_FolderName = "BinaryTrackRecordData";
        const string k_FileName = "BestTimeRecordData";
        const string k_FileExtension = ".json";

        public static void Add(TrackRecord record)
        {
            if (Sortedrecord.Count < BestTimeRecordSize)
            {
                Sortedrecord.Add(record.costTime, record);
            }
            else
            {
                Sortedrecord.RemoveAt(0);
                Sortedrecord.Add(record.costTime, record);
            }

        }

        public static void Save(BestTimeRecord bestrecords)
        {
            bestrecords.records = new List<TrackRecord>();
            foreach (TrackRecord sr in Sortedrecord.Values)
            {
                bestrecords.records.Add(sr);
            }

            string folderPath = Path.Combine(Application.persistentDataPath, k_FolderName);

            if (!Directory.Exists(folderPath))
                Directory.CreateDirectory(folderPath);

            string dataPath = Path.Combine(folderPath, k_FileName + k_FileExtension);


            if (File.Exists(dataPath))
            {
                Debug.Log("delete");
                File.Delete(dataPath);
            }
```

```
48              FileStream fileStream = File.Open(dataPath,
FileMode.Create);
49              byte[] bytes = new
UTF8Encoding().GetBytes(JsonUtility.ToJson(bestrecords));
50
51              fileStream.Write(bytes, 0, bytes.Length);
52              //每次读取文件后都要记得关闭文件
53              fileStream.Close();
54
55          }
56          public static BestTimeRecord CreateDefault()
57          {
58              BestTimeRecord defaultRecord = new BestTimeRecord();
59              defaultRecord.records = new List<TrackRecord>();
60              Sortedrecord.Capacity = BestTimeRecordSize;
61              return defaultRecord;
62          }
63
64          public static BestTimeRecord getBestTimeRecord()
65          {
66              BestTimeRecord br = Load();
67
68              Sortedrecord = new SortedList(new FloatSort());
69              Debug.Log(JsonUtility.ToJson(br.records));
70              foreach (TrackRecord record in br.records)
71              {
72                  Sortedrecord.Add(record.costTime,record);
73              }
74              return br;
75          }
76
77          public static BestTimeRecord Load()
78          {
79              string folderPath =
Path.Combine(Application.persistentDataPath, k_FolderName);
80
81              if (!Directory.Exists(folderPath))
82                  Directory.CreateDirectory(folderPath);
83
84              string dataPath = Path.Combine(folderPath, k_FileName +
k_FileExtension);
85              Debug.Log(dataPath);
86              if (!File.Exists(dataPath))
87              {
88                  return CreateDefault();
89              }
90
91              using (FileStream fileStream = File.Open(dataPath,
FileMode.Open))
```

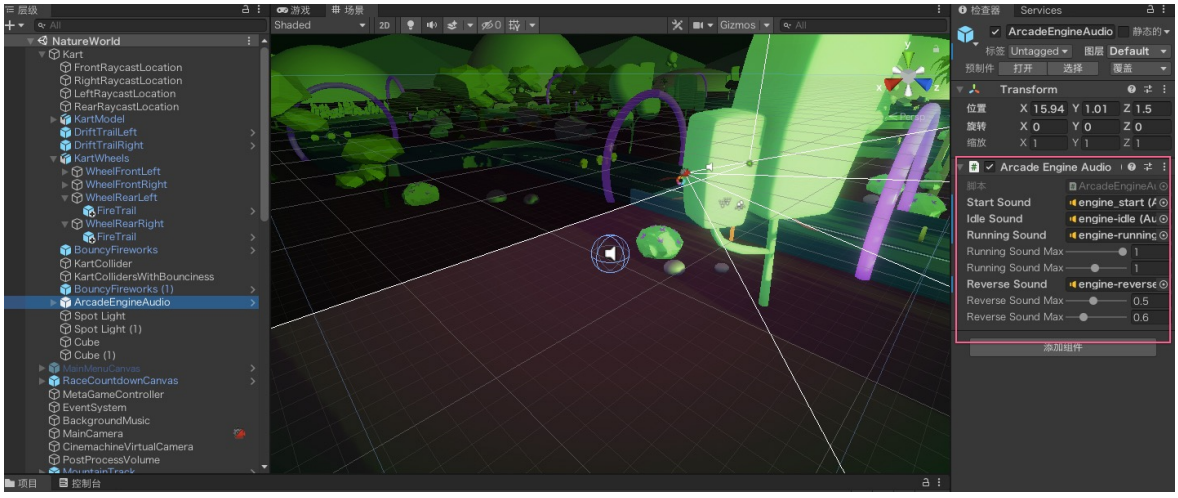
```
92              {
93                  if (fileStream.Length == 0)
94                      return CreateDefault();
95
96                  try
97                  {
98                      byte[] bytes = new byte[10000];
99                      fileStream.Read(bytes, 0, bytes.Length);
100                     //将读取到的二进制转换成字符串
101                     string s = new UTF8Encoding().GetString(bytes);
102                     BestTimeRecord loadedRecord =
JsonUtility.FromJson<BestTimeRecord>(s);
103
104                     if (loadedRecord == null)
105                         return CreateDefault();
106                     return loadedRecord;
107                 }
108                 catch (Exception)
109                 {
110                     return CreateDefault();
111                 }
112             }
113         }
114
115     }
```

可以看出主要是使用SortedList去进行记录的排序，当超过十个记录时会使用Sortedrecord.RemoveAt(0)删除

- 音效设计：（**已实现基本要求**）基于原来demo的基础上，顺利保证了兼容多人联网的各种音效的流畅效果。

  视频时间：游戏全程

  相关截图：



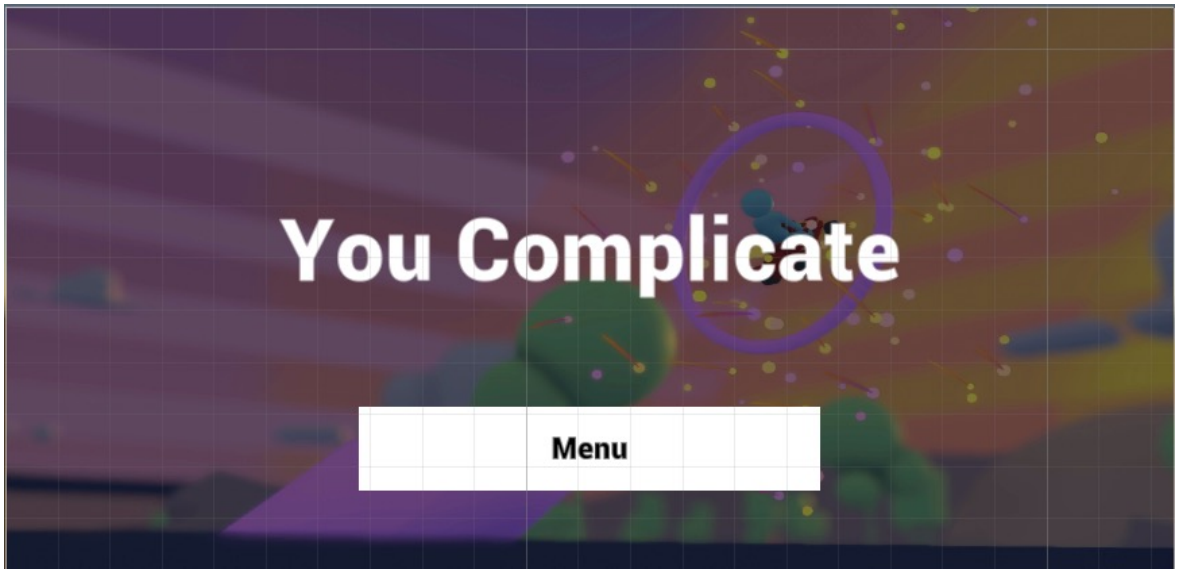
设计与实现：拖拽入相关音效即可，少量代码修改。如只需要产生音效的时候，使用下列代码：

```
AudioUtility.CreateSFX(CollectSound, transform.position,
AudioUtility.AudioGroups.Pickup, 0f);
```

- 完善UI：（**已实现高分要求**）（**基本要求**）：游戏基于多人模式完善了启动UI，关卡结算UI，选项的UI，并增加了房间匹配的UI，战绩排行榜查询的UI。（**高分要求**）除了完善与增加UI以外，还在原来demo基于wasd键盘控制的基础上迁移到了手机平台，增加了摇杆控制的UI，使得游戏能够在手机上顺利操作。

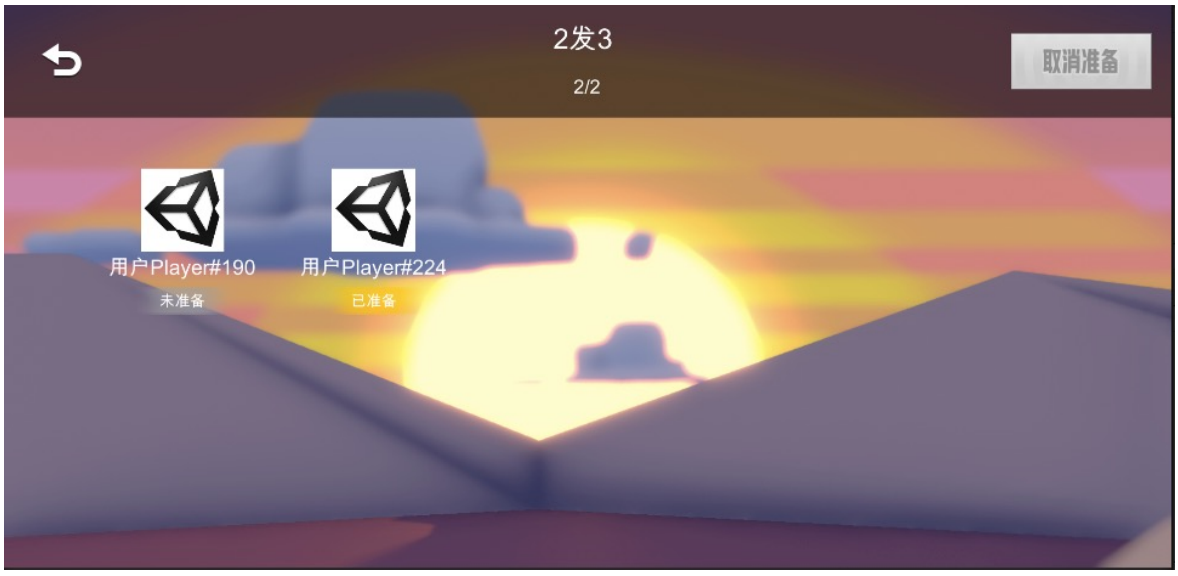
  视频时间：游戏全程

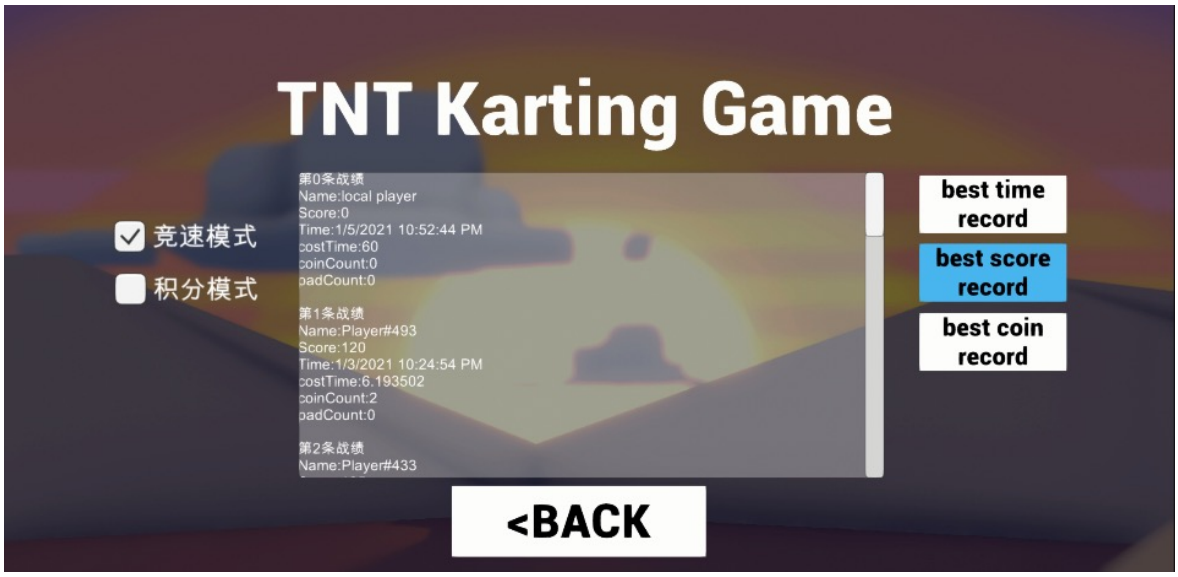
  相关截图：




游戏大厅
Player#779
登录


游戏大厅
自动匹配
创建房间
房间2发3
1/2


2发3
2/2
取消准备
用户Player#190
未准备
用户Player#224
已准备



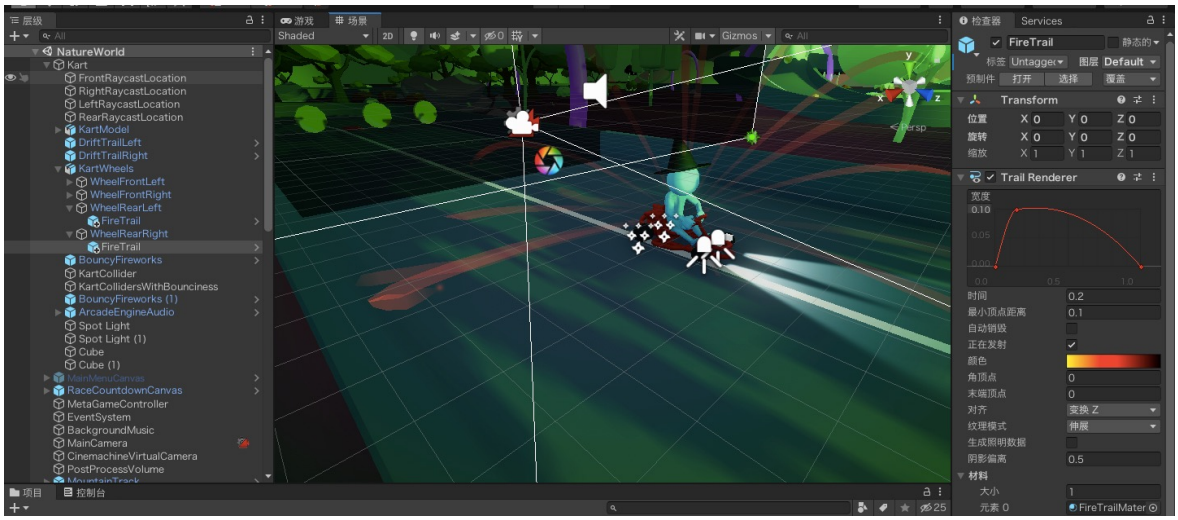
设计与实现：因为UI涉及元素过多，此处不一一赘述，特别只强调一下摇杆操作的实现。摇杆操作的代码在VariableJoystick类中实现。核心代码如下：

```
protected override void HandleInput(float magnitude, Vector2 normalised, Vector2 radius, Camera cam)
    {
        if (joystickType == JoystickType.Dynamic && magnitude > moveThreshold)
        {
            Vector2 difference = normalised * (magnitude - moveThreshold) * radius;
            background.anchoredPosition += difference;
        }
        base.HandleInput(magnitude, normalised, radius, cam);
    }
```

- 增加车辆的效果 （**已实现高分要求**） ： （**基本要求**） 增加车辆的光迹,增加轮子的粒子效果。 （**高分要求**） ： 通过材质和特效,给车辆增加了车灯,并实现了前车灯的"光效"。

视频时间：关卡自然世界通关视频.mp4 20s以后的游戏全程

相关截图：

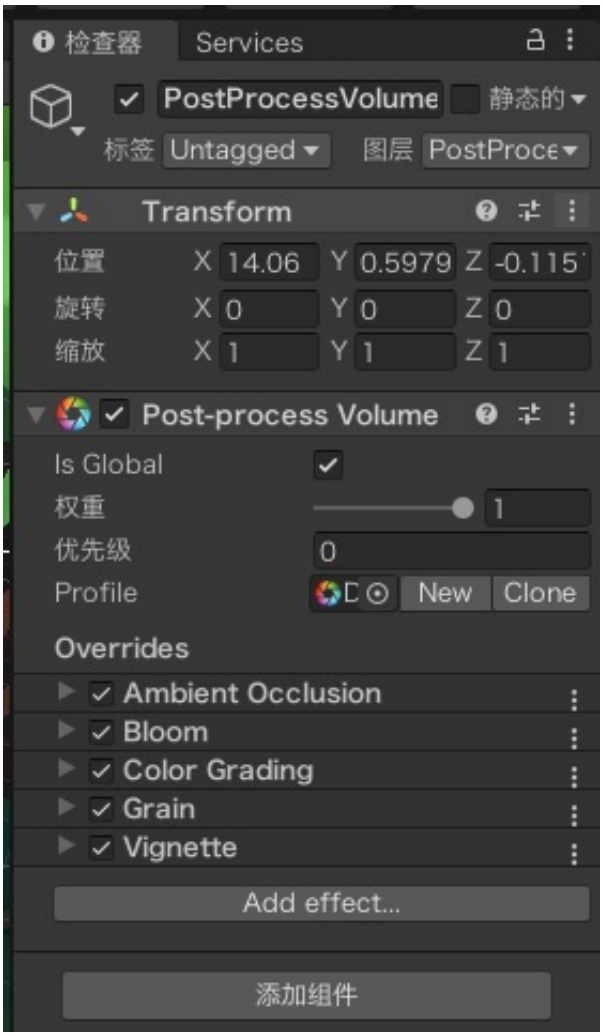


设计与实现：光迹根据教程拖入预制件即可。光源只需在Unity创建相关组件，放到对应位置即可，无需修改代码。

- 改变游戏的图像效果（**已实现基本要求**）：**（基本要求）**加入了动态光源车灯，增加了后处理效果来调节场景氛围。并根据不同的主题设置了不同的后处理效果。

  视频时间：关卡自然世界通关视频.mp4 20s以后的游戏全程

  相关截图：

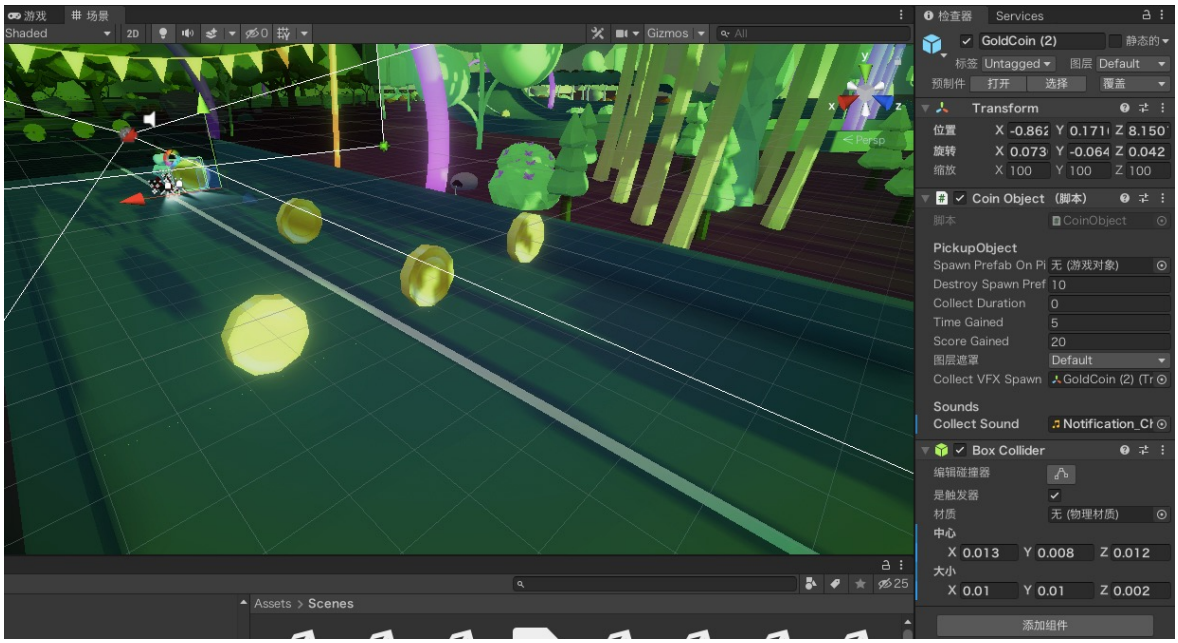


设计与实现：不同关卡根据教程使用不同的postprocess，进行微调即可

- 增加获取金币（**已实现高分要求**）：**（基本要求）**路面上有金币可以采集，碰撞后获取金币，并且金币可以增加玩家分数与比赛限制时间，同时金币也有自己的计数，获得的金币数量在游戏结束后也会同时记录在本地。可以在历史记录中查阅。并且有金币排行榜会储存金币获取数量最多的十次记录。**（高分要求）**在基本要求的基础上，因为涉及到多人联机，因此金币的逻辑就变得更为复杂了。而本游戏在双人的基础上进行了兼容，能够在一位玩家吃金币时，吃金币的结果只会影响到自己，同时把金币的结果对其他玩家进行同步，并且吃掉的金币会同时在所有玩家场景中消失。兼顾了多人联机的细节。

  视频时间：关卡自然世界通关视频.mp4 20s以后的游戏全程

  相关截图：



设计与实现：金币的实现代码主要在CoinObject和CoinManager类中，CoinManager主要负责金币数量的记录，CoinObjec则负责金币的相关功能，可以看到在触发碰撞体时，会去检测是否会自己的车碰撞到的，如果不是则只destroy金币，但是不会增加相应的时间，分数等，当然金币也有设置了相应的音效：

```
void OnCollect()
    {
        if (CollectSound)
        {
            AudioUtility.CreateSFX(CollectSound, transform.position, AudioUtility.AudioGroups.Pickup, 0f);
        }

        if (spawnPrefabOnPickup)
        {
            var vfx = Instantiate(spawnPrefabOnPickup, CollectVFXSpawnPoint.position, Quaternion.identity);
            Destroy(vfx, destroySpawnPrefabDelay);
        }



        TimeManager.OnAdjustTime(TimeGained);
        ScoreManager.OnAdjustScore(ScoreGained);
        CoinManager.OnAdjustCount(1);

    }

    void OnTriggerEnter(Collider other)
    {

        if ((layerMask.value & 1 << other.gameObject.layer) > 0 && other.gameObject.CompareTag("Player"))
```

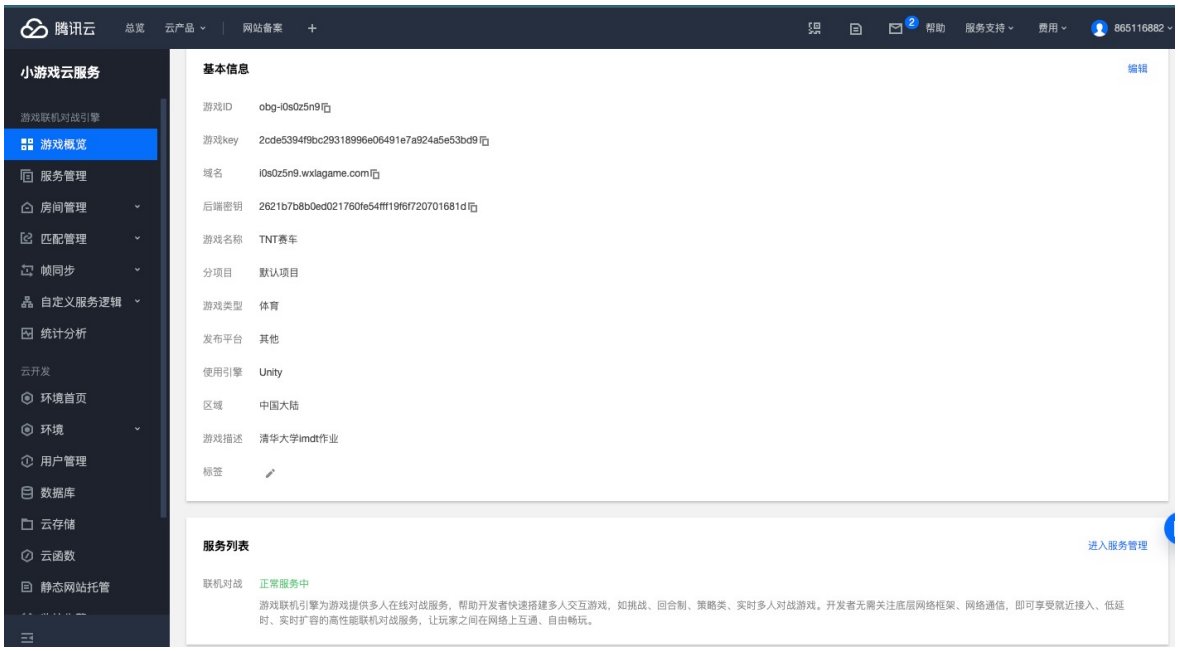
```
        {
            KartMovement km = other.GetComponent<KartMovement>();

            if (!km.IsOtherKart)
            {
                OnCollect();
            }

            Destroy(gameObject, collectDuration);

        }
    }
```

- 双人联机（**难度系数X5**）（**已实现高分要求**）：使用帧同步，已完成一个多人联机版本（1到多人都可运行），可以直接通过因特网连接，直接可用，相关服务已部署在腾讯云。双机都同时开局，比赛开始后双人都可以看到对方的赛车状态。HUD会显示各玩家单局最高分数，比赛结束后。可以看见自己的对局结果。

  视频时间：关卡自然世界通关视频.mp4 0-40s属于房间匹配 剩下为竞赛模式

  相关截图：

  设计与实现：联机功能基于腾讯云sdk（https://cloud.tencent.com/document/product/1038）开发。服务部署在腾讯云上，控制台如下所示：



本游戏实现了自动匹配，创建房间，快速加房等功能，具有非常完善的功能。相关功能及流程如图所示：

因为联机相关代码非常多，此处不一一赘述。联机的核心功能由Client类实现，包括Listener的创建，各种老师感兴趣可以看相关源码。此处只展示桢同步和数据传输的代码：

```
1  void TryBeginBattle () {
2          if (Global.Room != null) {
3              if (Global.Room.RoomInfo.PlayerList.Count !=
   Convert.ToInt32 (Global.Room.RoomInfo.MaxPlayers)) {
4                  return;
5              }
6              foreach (PlayerInfo player in
   Global.Room.RoomInfo.PlayerList) {
7                  if (player.CommonNetworkState ==
   NetworkState.CommonOffline)
8                      return;
```

```
                if (player.CustomPlayerStatus != 1)
                    return;
            }
            StartFrameSync ();
            isReadyToBattle = false;

        }
    }

    void StartFrameSync () {
        StartCoroutine (LoadKartScene ());
        // 开始帧同步
        Global.Room.StartFrameSync (eve => {
            try {
                if (eve.Code == ErrCode.EcOk) {
                    isStartFrameSync = true;
                    isInBattle = true;
                } else {
                    roomsPanel.setReadyBtn.interactable = true;
                }
            } catch (Exception e) {
                Debug.LogError (e);
            }
        });
    }
    void OnFrame (Frame fr) {
        if (game == null || !game.isStarted) return;
        if (!isRecvFrame && fr.Id > 10) {
            // Start countdown at the same time for all the clients
            game.raceCountdownTrigger.TriggerDirector ();
            isRecvFrame = true;
        }
        var acceleration = game.myKartInput?.Acceleration ?? 0;
        var steering = game.myKartInput?.Steering ?? 0;
        complicated = GameFlowManager.complicated;
        score = GameFlowManager.score;

        var para = new SendFramePara {
            Data =
            $"{acceleration},{steering},
{game.myKartInfo.Position.x:0.###},
{game.myKartInfo.Position.y:0.###},
{game.myKartInfo.Position.z:0.###},
{game.myKartInfo.Rotation.x:0.####},
{game.myKartInfo.Rotation.y:0.####},
{game.myKartInfo.Rotation.z:0.####},
{game.myKartInfo.Rotation.w:0.####},{complicated},{score}"
        };
        // 发送帧
```

```
        Global.Room.SendFrame (para, eve => { });

        foreach (var item in fr.Items.Where (item =>
game.otherKarts.ContainsKey (item.PlayerId))) {
            game.otherKarts[item.PlayerId].OnFrame ((string)
item.Data);
        }
    }
```

可以看到，在同步时会把加速，转向，三个位置数据，四个旋转角度数据，以及是否完成游戏，分数这些数据进行广播帧发送，这样只要在这个房间里的玩家，都会受到该数据。

对应的接受数据，进行处理的代码在MetaGameController.cs中，它会接收解析数据，并进行相应处理，代码如下：

```
public class OtherKart {
        public GameObject go;
        public NetworkInput input;

        KartMovement movement;
        Vector3 pos;
        Quaternion rot;

        public OtherKart (GameObject go, NetworkInput input) {
            this.go = go;
            this.input = input;
            movement = go.GetComponent<KartMovement> ();
            movement.IsOtherKart = true;

            pos = new Vector3 ();
            rot = new Quaternion ();
        }



        public void OnFrame (string str) {
            string[] vals = str.Split (',');
            if (vals.Length == 11) {
                int intAcc;
                int intSteer;
                int score;
                float px, py, pz;
                float rx, ry, rz, rw;
                bool complicated;
                if (Int32.TryParse (vals[0], out intAcc) &&
                    Int32.TryParse (vals[1], out intSteer) &&
                    float.TryParse (vals[2], out px) &&
                    float.TryParse (vals[3], out py) &&
                    float.TryParse (vals[4], out pz) &&
```

```
                    float.TryParse (vals[5], out rx) &&
                    float.TryParse (vals[6], out ry) &&
                    float.TryParse (vals[7], out rz) &&
                    float.TryParse (vals[8], out rw) &&
                    bool.TryParse(vals[9], out complicated) &&
                    Int32.TryParse(vals[10], out score)) {
                    input.m_Acceleration = intAcc;
                    input.m_Steering = intSteer;
                    pos.x = px;
                    pos.y = py;
                    pos.z = pz;
                    rot.x = rx;
                    rot.y = ry;
                    rot.z = rz;
                    rot.w = rw;
                    rot.Normalize ();

                    if (ScoreManager.bestScore < score)
                    {
                        ScoreManager.bestScore = score;
                    }
                    if (!GameFlowManager.complicated)
                    {
                        GameFlowManager.complicated = complicated;
                    }
                    movement.syncPosition = pos;
                    movement.syncRotation = rot;
                }
            }
        }
    }
```

因为时间原因，来不及做单局多人联机的排名，目前游戏结果目前只有三种，分别是win,complicate和lose，当时间到没有完成比赛时，游戏结果为Lose，当第一个到达终点时，结果为WIn，当时间限制内达到终点但是却没有成为第一名时，结果为complicate。当然就目前的功能来说，已经满足双人游戏需求了。

- 其他加分：赛道计分板显示当前用的时间等，时间显示会变化。玩法在一开始就可以进行模式选择等。玩家自己的分数以及本局最高分数也会同步显示在HUD上，并实时更新。

  视频时间：关卡自然世界通关视频.mp4 游戏开始即可看到

  相关截图：

设计与实现：模式选择通过toggle实现，积分模式通过判定积分大小而不是完成时间来决定游戏结果。

分数的管理主要由ScoreManager类实现，而相关的显示则由ScoreHUDManager类实现。在游戏中，每一帧都会比较收到的玩家分数与自己的分数是否比best Score大，如果是则进行替换。

心得体会

心得体会就是，一定要听从老师的劝告，老师说难度非常高，挑战请慎重的，就一定是这样的。不自量力的我想要挑战多人联机功能可谓是付出了超级多的时间成本，看着别的同学按照教程两三天就做完了，而且虽然没有联机功能但是表现炫酷。而我自己做联机功能没有直接的教程，只能自己实现，期间也遇到了不少坑，花了十多天才将将搞定，就这也有不少的瑕疵，不过最终还是比较满意的，至少遇到的bug都解决了，剩下的瑕疵也就是时间上的限制导致的细节的缺失，比如只做了多人游戏的结果显示（胜利，完成，还是失败），多人成绩表单式的排名没有时间做了。

做这个作业的过程中，遇到了很多同学没有遇到的坑，只能说任何一个简单的小功能，小需求，只要和通信联网相关就是一个复杂的问题，比如说在游戏开发的后期我发现游戏一周目没有问题，但是二周目会有问题，原因在于挂在联机脚本的对象是dont destroy on load的，而这导致了再来一盘时会有多个重复的联机脚本对象，导致出现问题（因为联机脚本需要从房间匹配到游戏结束之前都存在，

所以我设置为了dont destroy on load）。我解决的方法是在游戏结束返回主菜单的时候，去destroy之前的这个联机脚本对象，然而又出现问题了，原因在于腾讯云的SDK中Listener有bug，它在初始化的时候会用foreach去删除监听元素（注，foreach不能直接删除元素），做清理。也就是说Listener使用的是静态方法，只能初始化一次，第二次初始化是会出现问题，不能直接把之前的Listener干掉重新初始化。然后我又只能在初始化联机脚本对象时去判断Listener是否已经初始化过，如果初始化过就复用Listener。不幸的是这样依旧有问题，因为之前的状态没有清理，又得增加游戏结束退出房间，清理状态的代码。总的来说就是，每一个小问题解决了又会有新的小问题，特别是在做金币，分数功能时，更是要考虑到这些信息的多人同步，不同玩家的判断等等，开发工作量可以说翻倍了还不止。而且多人联机调试起来做的非常的麻烦...

在发完了牢骚以后，讲讲技术方面的心得体会吧。因为时间大部分花在了联机功能上，因此也会着重网络这一块。在做之前，我还是挺信心满满的，因为个人有个一段时间的大厂服务器开发经验，所以觉得多人联机应该不难，但老实说游戏里的桢同步网络通信和基于RESTful的WEB开发还区别很大。传统的web开发使用的是http协议，以短连接无状态为主要特点。只需要把数据过一遍业务逻辑即可，相互的交互并不多，端与端更是很少。而因为本游戏是基于腾讯云SDK开发，没有自己搭建服务器，而是通过通过腾讯云提供的服务，使用广播监听的方式，进行不同客户端的通信，服务端不储存任何信息，也不进行任何业务逻辑，可以说只是通过腾讯云实现的一个p2p的桢同步。如何在不同客户端处理状态和消息是一大难点，也是非常麻烦的事情，在这其中我也学到了很多知识。

自己摸索实现的除了多人联网还有记录储存，UI操作迁移和金币功能，积分模式，这几个也是没有看教程，自己完成的。如何在多人联网基础上实现上述功能也让我受益匪浅，使得我对C#脚本在unity中的应用得到了极大的锻炼。现在基本已经可以无障碍地在unity中进行C#脚本编程了。

而其他要素，如改变汽车形象，后处理，音效等等，都是我按照官方教程一步步做的，使得我对unity从陌生到熟悉。真正的了解了游戏的整个开发过程是什么样的，也是一次非常好的实践机会。

总的来说，多人联网，及其基础上的记录储存，UI操作迁移和金币功能，这四点是我开发工作量最大的四个部分，也是最有成就感的四个部分。而其他要素的实现也加深了我对游戏开发相关技术理解与运用，使得我更深刻地能够讲老师上课讲授的知识融会贯通，是一次非常难得的锻炼机会。