2023年第四届四川省大学生智慧文旅作品创新创作大赛

作品类型：科技创新设计类

作品名称：熊猫跑酷

学生姓名：程文林 陈咏琪 肖玮 李逍遥

指导教师：郭晓丹

参赛学校：成都锦城学院

熊猫跑酷

摘要：大熊猫一直是我国的国宝，很多人想要见到他，但却没有一款专属于它的游戏。因此该项目是一款以成都文化为背景的特色跑酷游戏。主要涉及成都特色国宝熊猫以及大运会，主要游戏方式为大熊猫需要躲避饲养员的追逐并在逃跑的跑道上不断吃金币补充体力，该项目主要通过：大熊猫的形象、成都的建筑，结合互联网、大数据技术开发的休闲小游戏。让用户在休闲的同时了解大熊猫及成都。

关键词：成都；休闲；小游戏

目 录

第一章 需求分析

1.1 项目简介

休闲、益智类游戏。背景为大运会来临，可爱的大熊猫也想踢着它的小足球坐地铁去看大运会，偷偷跑出动物园，被饲养员发现，然后在地铁上追赶的游戏。玩家扮演想要去看大运会的大熊猫，大熊猫逃出动物园，大门关上，追逐开始。大熊猫需要躲避饲养员的追逐并在逃跑的地铁上不断吃竹子补充体力。熊猫酷跑每一局中都会带你领略到独属于成都的蓉文化，特色的成都地铁、特色的成都风景，在你眼中大放光彩，以绚烂美丽的一面刻印在你的脑海中。大家在聚会、下课、坐地铁、休闲在家，都可以随时随地来一局，轻松挑战历史记录，体验突破自我的快感。

1.2 项目总体目标

搭建属于成都特有文化的场景，创建出大熊猫的角色形象，两者相互结合创建出的 3D 酷跑。最终实现用户不仅能快乐的玩耍，还能领略到成都大运会的风采和激动的心情。

1.3 可行性分析

1.3.1 市场需求

目前，休闲益智类的小游戏仍受大家的喜爱，不管是青少年、小朋友、老年人都占有一定的比例。我们在熊猫酷跑中还融入成都独有的文化和风格，我们以成都地铁为背景搭建的 3D 场景，有一定宣传成都美丽文化的作用。在我们游戏市场中是一处新的风景。因此有一定的市场需求价值。

1.3.1 技术可行性分析

通过 Unity、VS 软件编写 C#代码创建场景类、人物类、计时类产生 。

1.3.2 竞争环境

市场上已存在很多类似于酷跑的小游戏，竞争环境压力较大。在和熊猫酷跑类似的游戏中，我们与成都的美丽相结合，重点在于突出成都大运会的风采，想要一起乘坐成都地铁的

游客们，能让大家在玩耍中收获快乐，能领略到我们成都这座天府之国的魅力；熊猫酷跑可以单机游玩，让大家随时随地的进行玩耍，一局时间大概 5 分钟左右，且具有暂停的功能。可以随时随地的退出。因此，在竞争环境较大的情况下，我们仍有属于自己的优势。

1.4 需求规定

1.4.1 竞争环境功能需求

（1）搭建地图场景

（2）设计形象可爱的大熊猫并生成

（3）角色体力消耗计时显示

（4）角色需转换视角

（5）角色可实现前后左右的基本移动

（6）角色可以实现跑、跳

如图：

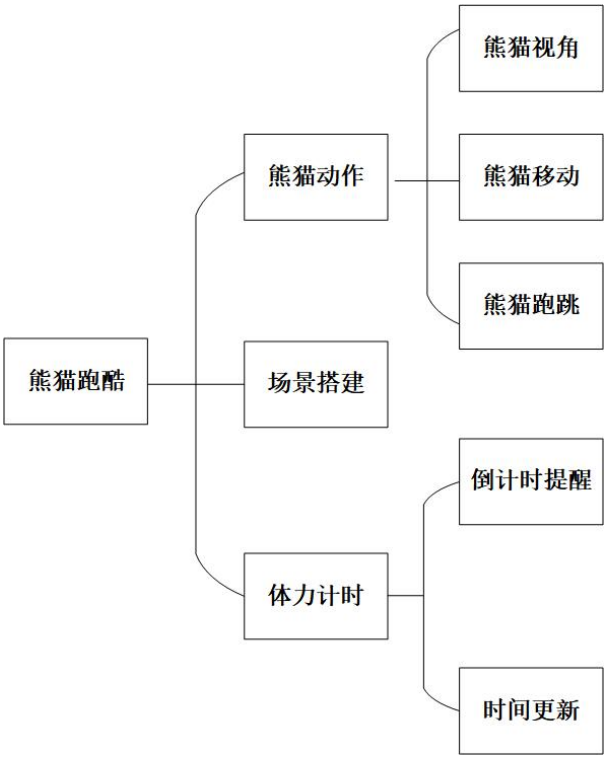

图 1.熊猫跑酷总体功能

1.4.2 性能需求

随着游戏行业的发展，游戏数量越来越多，玩家数量也越来越多，因此玩家面临的选择也越来越多，所以我们对游戏的内容和质量进行了一系列的要求。

通过 Unity，VS 软件编写 C#代码对成都地铁进行搭建，创建 3D 角色的大熊猫。以美丽的游戏界面来留住我们的玩家。

熊猫酷跑需要高帧率（60fps）或者是（120fps）进行画面的渲染，以保证游戏画面的流程。

物理碰撞处理：游戏中的角色、障碍物和道具都需要进行实时的物理碰撞检测，以确保游戏的逻辑的正确。

数据的处理：熊猫酷跑需要对大量的游戏数据进行处理，包括熊猫的位置、速度、碰撞检测结果等。

网络通信问题：在联机的模式下，我们也尽可能的降低网络延迟带来的影响。

1.4.3 非功能需求

（1）稳定性要求

熊猫酷跑的稳定性我们分为两方面：

一方面是游戏自身的稳定性，我们要求整体游戏的画面是流畅的，操作相应反馈及时，且在酷跑的过程中游戏逻辑是正确的，例如：吃金币、吃道具等后的反映，不会出现不合理的游戏逻辑。

另一方面是成都地铁轨道的稳定性，包括地铁轨道的平滑程度以及轨道对玩家进行弹跳动作时，给予一定的反馈。

（2）环境

①游戏系统环境

有官方客服，进行实时的玩家意见收集，最后我们将把对玩家的反馈，反映在下一次的更新版本中。

可以支持手机、平板等移动设备进行玩耍。

游戏难度会随着玩家越跑越远而速度也越来越快。

游戏排名：会记录玩家跑的最远的一次记录。再根据后续玩家打破记录后再进行实时的更新。

游戏元素包括：大熊猫、成都地铁、障碍物、金币、道具等。

②游戏运行环境

我们熊猫酷跑通常都可以在移动设备下正常运行。例如：安卓系统，iOS 系统，鸿蒙系统环境下都可以正常的进行运行。

第二章 概要设计

本游戏主要使用的技术是后端技术，其中主要包括 unity、c#。本游戏是一个简易的休闲小游戏，暂时没有进行安全设置。我们通过梳理游戏大致结构，再分功能编写代码，最后加上特效，再次修改代码，经过不断调整，终于将项目完成。

2.1 逻辑框架设计

在游戏开始设计的时候，我们参考了同类型游戏的设计思路，并经过团队成员的讨论，大致对项目的逻辑框架进行了梳理。项目主要分为 MVC 模块设计、音乐管理模块以及跑道轮换三个模块，具体板块见下图。

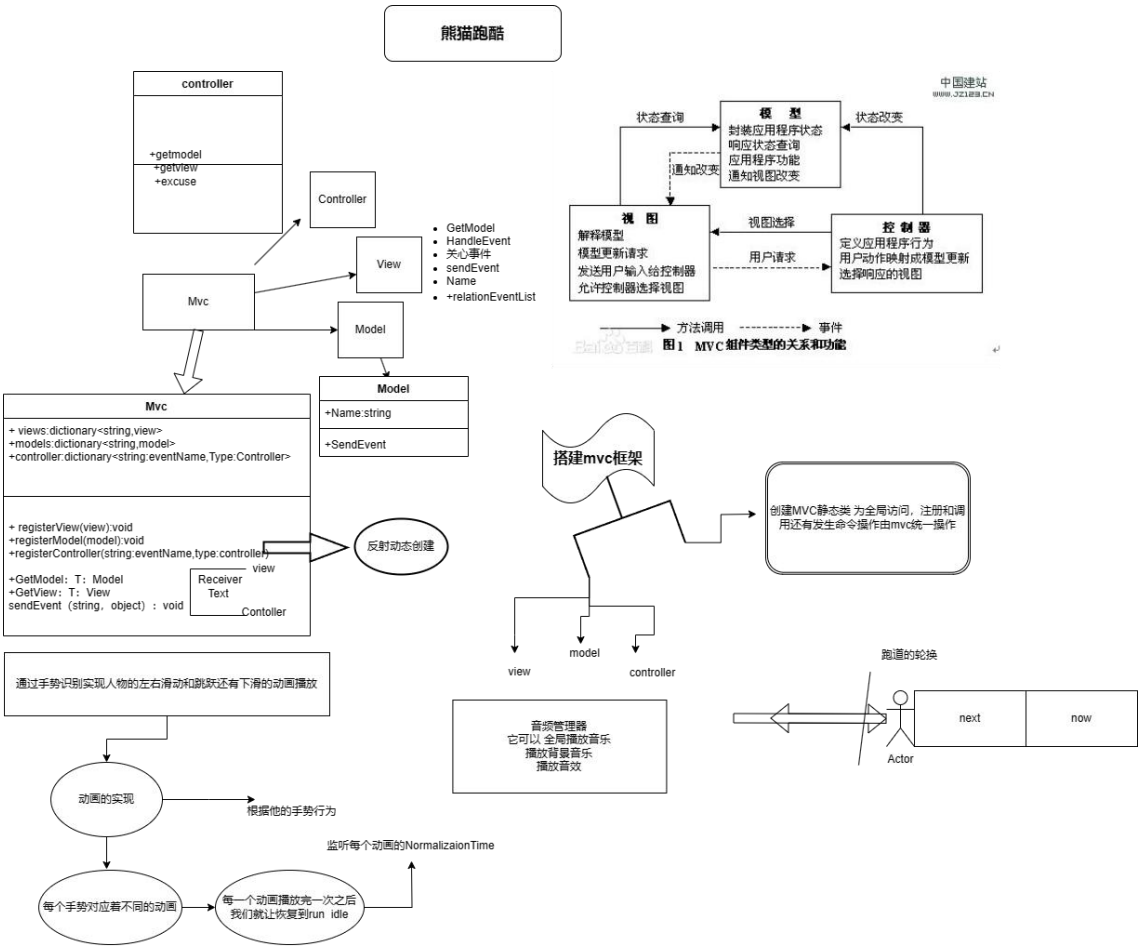

图 2.逻辑框架设计图

2.2 代码编写

在编写代码的时候，我们首先是对 MAC 模块进行编写，搭建项目基本框架。再将角色与跑道拖入 unity 中进行位置调整，使角色最开始的位置处于跑道中间，同时调整相机，使游戏界面的视角位于角色正后方，方便游戏过程中观察路障，从而给出正确的行动。

第三章 详细设计

3.1MAC 模块

MVC 模块又分为三个部分，分别为控制器（Controller）、模型（Model）和视图（View）。在控制器的设计上，我们搭建了基本的控制器框架。在此基础上我们创建了 EnterSceneController 类，其中最为重要的是 LoadGameScenesEvent()函数，它主要用于初始化游戏场景，激活游戏对象，设置模型状态，注册视图，生成游戏中的物体，并触发游戏事件，从而加载游戏场景。

```
private void LoadGameScenesEvent()
{
    GameManager.Instance.SetActiveGameObject(GameObject.FindWithTag(Tags.GamePanel));
    GameModel gm = GetModel<GameModel>();
    gm.IsPause = false;
    gm.IsPlay = true;

    RegisterView(GameObject.FindObjectOfType<RunningGameView>());

    GameManager.Instance.SetActiveGameObject(GameObject.FindWithTag("Player"));
    RegisterView(GameObject.FindWithTag(Tags.Player).GetComponent<Player>());

    GameObject.Find("Canvas").transform.Find("UIPausePanel").gameObject.SetActive(true);
    RegisterView(GameObject.FindObjectOfType<UIPauseView>());

    GameObject.Find("Canvas").transform.Find("ContinuePanel").gameObject.SetActive(true);
    RegisterView(GameObject.FindObjectOfType<ContinueView>());
    GameObject go = GameManager.Instance.objectPool.Spawn("Coin");
    go.transform.position = new Vector3(0, 0, 80f);
    GameObject go1 = GameManager.Instance.objectPool.Spawn("Coin");
    go1.transform.position = new Vector3(2, 2, 85f);
    GameObject go2 = GameManager.Instance.objectPool.Spawn("Coin");
    go2.transform.position = new Vector3(0, 2, 80f);
    GameObject go3 = GameManager.Instance.objectPool.Spawn("Coin");
    go3.transform.position = new Vector3(0, 0, 85f);
    GameObject go4 = GameManager.Instance.objectPool.Spawn("Coin");
    go4.transform.position = new Vector3(0, 0, 130f);
    GameManager.Instance.objectPool.Spawn("Item_Magnet").transform.position = new Vector3(0, 0, 60);
    Arg_GameTime argTime = new Arg_GameTime() { timer = 60f,totalTime=60f,isInitTime=true };
    Mvc.SendEvent(Consts.E_UpdateTime, argTime);
}
```

图 3. 控制器板块部分代码截图

在视图的设计上，我们首先对角色功能进行了编写，完成了角色的基本移动功能。在角色设计中，我们使用了 FixedUpdate()函数，用于处理物理相关的更新。它在固定的时间间隔内执行，通常用于处理角色控制、物理模拟和其他需要固定更新频率的操作。

```
321     }
322     public void FixedUpdate()
323     {
324         if (GameManager.Instance.SearchPlayState()==false) return;
325         if (character.isGrounded)
326         {
327             if (y_velocity < -0.5f)
328             {
329                 if (isReduce==false)
330                 {
331                     runSpeed = recordSpeed;
332                 }
333                 //runSpeed = baseSpeed + (Z_distance / eachDistance);
334                 tempDistance += runSpeed * Time.deltaTime;
335                 if (tempDistance>=EachDistance)
336                 {
337                     tempDistance = 0;
338                     runSpeed += increamentSpeed;
339                     recordSpeed = runSpeed;
340                 }
341             }
342         }
343             y_velocity -= 12f * Time.deltaTime;
344         character.Move((Vector3.forward * runSpeed + Vector3.up * y_velocity) * Time.deltaTime);
345         Z_distance +=( runSpeed * Time.deltaTime);
346     }
```

图 4.Player 特殊板块设计截图

其次，进行视图框架的搭建用于创建视图对象，注册感兴趣的事件，与控制器进行通信，处理事件，并与模型进行交互，从而显示游戏中的信息和界面元素。

图 5.视图界面显示图

同时，在角色类中，我们对角色的动作进行了编写设计，分别为左划——左移、右划——右移、上划——跳跃、下划——下滑，使得角色在游戏过程中基本可以实现基础的行动。

图 6.角色移动显示图

在模型的设计上，首先搭建了一个简易的模型框架，其次创建一个派生类 GameModel，

使其继承于 Model 类，从而管理游戏的基本状态和数据，包括金币数量、总距离、道具状态等。

利用 UpdateCoinAndDistance()函数记录更新金币数量和总距离，同时通过不同的 bool 函数对游戏的起止、是否使用道具进行判断。

```
00    public void UpdateCoinAndDistance(Arg_CoinAndDistance arg)//用于更新金币数量和总距离
01    {
02        CoinAmount += arg.coin;//把当前游戏running时的吃到的金币加到总的游戏数据中
03        TotalDistance = arg.distance;
04    }
05    private bool isMegnet;//游戏中是否启用了磁铁道具
06    private int coinAmount; //表示游戏中的金币数量
07    private float totalDistance;//表示游戏中的总距离
08    private bool isPause;//表示游戏是否处于暂停状态
09    private bool isPlay=true;//表示游戏是否正在进行中
10    private float timer;//表示游戏的计时器
```

图 7.模型板块部分代码截图

3.2 音乐管理模块

在设计好 MVC 模块后，首先构建一个 GameManager 类，从而管理游戏的开始和结束。其次创建一个 Sound 类，对游戏的背景音乐及动作特效进行设置，使用了 Unity 引擎提供的标准方法和属性来处理音频源的初始化和设置。

图 7 为 Sound 类中的亮点，该句的意思是检查加载的音频资源 clip 是否存在（不为 null），并且它是否与当前背景音频源 bgAudio 中的音频剪辑不同。这是为了避免在播放相同音频时多次启动。

```
if (clip!=null&&clip!=bgAudio.clip)
{
    bgAudio.clip = clip;
    bgAudio.Play();
}
```

图 8.Sound 类部分截图

3.3 跑道轮换

首先创建一个 RoadChange 类，从而使游戏开始时生成并设置两个道路对象的初始位置，为随机生成跑道做准备。其次在该类中加入触发器方法无限生成并重复使用跑道，实现跑道轮换的效果，同时编写 OnTriggerEnter()函数处理一种无限道路生成的游戏中的逻辑，当玩家的某个物体与道路碰撞时，它会生成新的道路模块并将当前道路模块替换为新的道路模块。

这可以用于创建无限滚动的道路，以提供游戏的可玩性。

```
18      private void OnTriggerEnter(Collider other)
19      {
20          if (other.tag==Tags.Road)
21          {
22              int index = Random.Range(1, roadNum);
23              GameObject go = GameManager.Instance.objectPool.Spawn(relativePath + index);
24              go.transform.position = nextPattern.transform.position + Vector3.forward*160f;
25              GameManager.Instance.objectPool.UnSpawn(nowPattern);
26              nowPattern = nextPattern;
27              nextPattern = go;
28
29          }
30      }
```

图 9.跑道轮换截图

第四章 测试报告

测试用例ID	标题	前提条件	测试步骤	预期结果	优先级	测试结果	备注（含软件缺陷 ID）
1	“熊猫跑酷”游戏打开	1.网络正常 2.文件齐全	1.下载熊猫跑酷安装包 2.点击熊猫跑酷图标	1.成功打开安装包，进入游戏界面	1	通过	
2	角色左右移动实现	1.网络正常 2.文件齐全	1.点击熊猫跑酷图标 2.用手指左划、右划	1.成功打开安装包，进入游戏界面，显示完整 2.角色向左移或向右移	3	通过	
3	角色上下跳跃移动实现	1.网络正常 2.文件齐全	1.点击熊猫跑酷图标 2.用手指上划、下划	1.成功打开安装包，进入游戏界面，显示完整 2.角色向向上跳或向下倒	2	通过	
4	背景音乐显示效果	1.网络正常 2.文件齐全	1.点击熊猫跑酷图标	1.成功打开安装包，进入游戏界面，显示完整 2.听见背景音乐	2	通过	
5	动作音效显示效果	1.网络正常 2.文件齐全	1.点击熊猫跑酷图标 2.用手指左划、右划、上划、下划	1.成功打开安装包，进入游戏界面，显示完整 2.在滑动时听见音效	2	通过	

第五章 项目总结

5.1 项目协调

关于项目的协调，我们团队由四名成员构成，其中两名同学主要负责界面的动画效果设计以及角色的动作实现的技术问题；第三名同学主要负责界面的设计，排版布局、美化、以及填充文案部分；第四名同学主要负责整个项目实行中的素材收集以及开发文档编写工作。

5.2 项目难点

作品制作的创新点主要在于通过跑酷游戏的方式让用户更愿意了解成都，大家在聚会、下课、坐地铁、休闲在家，都可以随时随地来一局，轻松挑战历史记录，体验突破自我的快感。

在制作过程中，主要面对的困难是，如何充分展现成都的魅力，以什么样的形式让用户感兴趣去了解。通过资料的查找，确定符合主题的素材，相应游戏的浏览，用户的调研最终确定展示内容和吸引用户的方式。然后就是在收集制作页面所需素材，如背景，场景，角色，道具等，经过反复考量，选出了以熊猫为原型进行角色设计。同时还注意颜色的搭配，在游戏的设计上是贴近众多用户的，游戏简单符合大众，不断测试，一步步实现。

相对于其他跑酷游戏，熊猫跑酷更具推广性。首先界面美观，界面的配色，背景音乐都经过反复推敲。其次是运用了熊猫的角色设计，让人进一步了解成都的特色，最后就是跑酷游戏的设计，让用户产生了解的兴趣，且让用户有更深刻的体验。