1. **游戏概述**

Runningman是一款传统的跑酷游戏。看似美丽的小岛实则暗藏杀机，主人公需要永不停歇的奔跑才能从小岛逃生。

跑酷游戏以其简单的剧情、便易的操作，但是经典的游戏奖惩机制设计，成为经久不衰的一种游戏类型。从冒险岛、超级玛丽，到现在层出不穷的画风精美，甚至3D跑酷游戏的出现，这种游戏类型牢牢地占据了市场一角。

1. **开发环境**

* Unity3D C#
* Android SDK

1. **程序分工**
2. **游戏介绍**

**4.1游戏界面描述**

游戏界面如视频所示，白色的人物是主人公，为了逃出恐怖小岛，他正在不停地奔跑。

游戏中央是一道道的路面，路面与路面之间可能会有悬崖，掉下去就是万丈深渊啦。

除了万丈深渊需要提防外，道路中的灌木丛可能是有毒的，如果碰到，也会一命呜呼哦。

不过在这紧张的逃命中，小岛的道路上也有随机掉落的金币可以采集哦。

在游戏界面的最顶端，HP是生命值，Score是得分。

**4.2游戏规则与功能介绍**

1. 人物控制

点击屏幕左侧，人物跳跃（用于越过障碍物、悬崖、捡金币）。

连续点击屏幕右侧，人物加速。

说明：人物只能连续跳跃两次。两次以上连续点击无效。

1. 游戏规则

死亡条件：碰触到灌木、掉下悬崖。

捡金币：碰触金币得分。

说明：人物有三条性命，每次死亡后，若PH值没有用光，则复活，从屏幕上方掉落，但是score分数清零。若PH值已经触底，则游戏结束。

1. 其他说明

背景与地面是无限生成滚动的，地面上的灌木与金币等地形是随机生成的。

只要没有死亡就不会停止游戏。

1. **主要代码分析（代码文件在/Assets目录下）**

**\*\*\*【开发语言：C#】\*\*\***

**背景控制**

|  |
| --- |
| 实现目标：   1. 实现背景的滚动。 2. 背景需要无限循环出现。 3. 当人物死亡时，背景需要静止。 4. 接收到加速信息时需要加速滚动（实现人物加速的效果）。   基本思路：   1. 背景的滚动通过素材的左移实现。 2. 背景的循环出现可以通过两张背景交替循环。当背景A左移出视图时，背景B出现在视图中，此时将A移动到B右侧，则当B移出时，A出现。 3. 人物死亡时，不更新背景位置坐标。 4. 加速信号到达时，背景坐标移动的距离增大。 |
| 背景的组件设置 |
| public class bg1control : MonoBehaviour {  public float speed = 1.0f;// 背景移动的速度  GameObject player;// 游戏对象：玩家  // 初始化时调用  void Start () {  player = GameObject.Find("Player");// 获取玩家对象  }    // 每一帧更新时调用  void Update () {  Vector2 v = transform.localPosition; // 获取背景的坐标（相对）  v.x -= (speed \* Time.deltaTime); // x坐标左移（实现背景的滚动效果）  if (Input.GetKeyDown(KeyCode.F)) // 得到加速信号  {  v.x -= 2\*(speed \* Time.deltaTime); // 加速滚动  }  if (v.x <= -53.7f) // 如果背景已经离开视图界面  {  v.x = -15.7f;// 则移动到第二张背景后（实现无限循环的滚动效果）  }  if (player.transform.localPosition.y > -10f)// 如果人物没有死亡  transform.localPosition = v; // 更新坐标  // 如果已近死亡，背景静止（不更新坐标）  } |
| Bg2的控制代码基本相似。 |

**地面控制**

|  |
| --- |
| 实现目标：   1. 实现地面的滚动。 2. 实现地面的无尽循环。 3. 每次出现的地面需要随机。 4. 人物死亡时背景的静止。 5. 加速信号下加速滚动。   基本思路：   1. 无限滚动与循环出现的思路同背景的处理。（通过素材的位置坐标处理实现）。 2. 死亡时的静止与加速信号的处理也与背景控制思路相似。 3. 每次出现的地面随机可以这样实现：两个地形组，每个组拥有四个地面。两个地形组循环出现，以此实现无尽地形。在A地形组离开视线循环到B后时，更新A的四个地面，四个地面从预设体中随机选择生成，（调整新的地面的位置等信息），成为新的地形。 |
| Ground物体的组件设置    地面的预设:  需要注意：地面需要增加刚体组件。实现和人物的承载和碰撞关系。 |
| public class ground1control : MonoBehaviour  {  GameObject[] grounds;// 地面预设体池  int grnum = 6; // 预设体数目  public float speed = 2.0f; // 地面移动速度  GameObject player; // 玩家  // 初始化时调用  void Start()  {  // 获取资源中的所有预设体  grounds = new GameObject[grnum];  grounds[0] = (GameObject)Resources.Load("Prefabs/gr1");  grounds[1] = (GameObject)Resources.Load("Prefabs/gr2");  grounds[2] = (GameObject)Resources.Load("Prefabs/gr3");  grounds[3] = (GameObject)Resources.Load("Prefabs/gr4");  grounds[4] = (GameObject)Resources.Load("Prefabs/gr5");  grounds[5] = (GameObject)Resources.Load("Prefabs/gr6");  // 获取玩家对象  player = GameObject.Find("Player");  }  // 每一帧更新时调用  void Update()  {  // 地图移动  Vector2 v = transform.localPosition; // 整个画面地形的位置坐标  v.x -= (speed \* Time.deltaTime); // 移动  if (Input.GetKeyDown(KeyCode.F)) // 加速信号  {  v.x -= (speed \* Time.deltaTime); // 实现加速  }  // 新地形生成  if (v.x < -45.7f) // 当地面移动出视图时，生成新地图  {  GroundGenerator();  }    // 新地形循环  if (v.x < -45.7f) // 新地形循环到接下来出现  {  v.x = 43.4f;  }  if (player.transform.localPosition.y > -10f)// 如果玩家已经死亡，停止  transform.localPosition = v;  }  } |
| // 地形生成（随机地形）  /// <summary>  /// 一个地形由四个随机的地面预设体组成。一共6个预设体（可增加），预设体有不同的位置、金币、灌木。  /// </summary>  void GroundGenerator()  {  GameObject newgrd; // 新地面对象  Vector2[] pos;// 坐标数组  int len = transform.childCount; // 地形中的地面个数  pos = new Vector2[len];  int i;  i = 0;  // 旧地面的销毁  foreach (Transform grdchild in transform) // 地形中所有的现有地面子物体  {  pos[i] = grdchild.localPosition; // 记录下他们当前的位置信息  Object.Destroy(grdchild.gameObject, 0f); // 销毁该物体  i++;  }  // 新地面的生成  for (i = 0; i < len; i++)  {  newgrd = Instantiate(grounds[Random.Range(0, grnum)], transform) as GameObject;// 实例化新的地面（随机）作为当前地形的子物体  newgrd.transform.localPosition = pos[i];// 将该新地面的位置设置为适当位置  newgrd.transform.parent = transform; // 设置新地面为当前地形的子物体  newgrd.transform.SetSiblingIndex(i); // 设置地面的index  }  } |
| Ground2 的控制代码基本相似。 |

**人物控制与游戏的进程**

|  |
| --- |
| 实现目标：   1. 人物的动作。（行走，跳跃，死亡）。 2. 加速信息与跳跃信息处理。 3. 人物触发金币、数木等游戏条件（游戏规则的实现）。 4. 人物死亡的判断。 5. 人物连跳次数的限制。   基本思路：   1. 人物的动作需要三个动画资源来实现，动画资源的播放迁移条件是需要由触发器来完成的。   迁移关系管理图如下：     1. 人物的组件如下   除了刚体、碰撞体组件在之外，还有动画播放与音效播放组件。 |
| public class playercontrol : MonoBehaviour  {  public int HP = 3; // 生命值  private int jumpCount = 0; // 跳跃的计数  public int score = 0; // 分数  private bool isjump = false; // 是否落地（关于跳跃计数清零）  private Animator playerAni; // 动画播放器  private AudioSource audio; // 音效播放器  private Rigidbody2D rbody; // 刚体组件  private Vector2 v; // 位置信息  public GameObject player; // 游戏物体：玩家  // 初始化  void Start()  {  rbody = GetComponent<Rigidbody2D>();// 初始化物体刚体组件  playerAni = GetComponent<Animator>(); // 初始化动画播放器（人物）  audio = GetComponent<AudioSource>();// 初始化音效播放器  }  // Update is called once per frame  void Update()  {  v = transform.localPosition; // 位置获取  if (HP < 0) // 若生命为0，游戏结束  {  Die(); // 调用死亡处理  v.y = -10f; // 将人物的位置定位在视图以下  score = 0; // 分数清零  }  else if (v.y <= -7f) // 生命不为0，但是人物触发死亡条件  {  HP--; // 生命数减少（若减少后小于0，下一帧就会被检测出，所以不需要格外检验）  v.y = 10f; // 人物更新到画面上方掉落  Die(); // 调用死亡处理  score = 0;// 分数清零  }  if (Input.GetKeyDown(KeyCode.J))// 接受到跳跃信息  {  Jump();// 跳跃  }  if (Input.GetKeyDown(KeyCode.F)) // 接受到加速信息  {  v.x += 4.0f \* Time.deltaTime; // 加速  }  transform.localPosition = v; // 更新位置坐标  } |
| 人物跳跃与死亡的处理  void Jump()// 人物的跳跃  {  jumpCount++; // 跳跃的计数  if (jumpCount < 3 && HP >= 0) // 如果连跳次数在允许范围内  {  rbody.AddForce(Vector2.up \* 380);// 实现跳跃  playerAni.SetTrigger("StartJump"); // 播放跳跃动画  AudioClip clip = Resources.Load<AudioClip>("jump");//加载跳跃音效  audio.PlayOneShot(clip); // 播放音效  }  }  void Die()// 人物死亡  {  playerAni.SetTrigger("Die");// 播放死亡动画  AudioClip clip = Resources.Load<AudioClip>("die");// 播放死亡音效  audio.PlayOneShot(clip);// 播放音效  } |
| 碰撞到金币与树木的代码处理  void OnCollisionEnter2D(Collision2D colli)// 关于碰撞  {  if (colli.collider.tag == "ground")// 如果检测碰撞到地面  {  jumpCount = 0;// 连跳的计数清零  }  else if (colli.collider.tag == "tree")// 如果检测碰撞到数木  {  if (HP < 0)// 触发死亡条件  {  Die();  score = 0;  v.y = -10f;  }  else  {  HP--;  Die();  score = 0;  v.y = 10f;  }  transform.localPosition = v;  }  }  void OnTriggerEnter2D(Collider2D colli)// 金币触发  {  if (colli.tag == "coin")// 碰撞对象为金币  {  score++; // 分数增加  Object.Destroy(colli.gameObject); // 金币自动销毁  }  }  } |

**提示信息**

|  |
| --- |
| 实现目标：   1. 显示玩家的姓名与分数   实现思路：  需要通过画布、文字、事件触发组件完成。 |
|  |
| 文字的更新也需要脚本控制：  public class textcontrol : MonoBehaviour {  GameObject player;// 玩家物体  Text txt;// 文本组件  // Use this for initialization  void Start () {  player = GameObject.Find("Player");  txt = GameObject.Find("Canvas/ScoreText").GetComponent<Text>();  }    // Update is called once per frame  void Update () {  playercontrol pla = (playercontrol)player.GetComponent(typeof(playercontrol));  txt.text = "HP:"+pla.HP.ToString()+"|Score:"+pla.score.ToString();// 更新信息  }  } |

游戏还有一些其他的设置与控制，但与上述基本相似，在这里就不一一列举了。

1. **游戏截图**

**游戏的运行效果录制在视频中。**

1. **心得、感想**

这个大程原本是组队完成，结果最后由于和队友之间的一些分歧所以自己仓促间独自完成，所以虽然已经尽可能让游戏部分完整，但是并非一个完整的应用。

不过尽管有这些遗憾，但是从这个大程和本学期的课程中，我还是学到了很多，关于终端应用开发的知识、关于游戏开发的知识，甚至新学会了C#语言。对于游戏开发的引擎unity也初步熟悉了操作，这款软件其实功能十分强大，不过在我手中只发挥了基础的作用，这是后话了。

最后要感谢老师的辛劳与教诲之恩。