南开大学 计算机大类

姓名 陈灿宇

学号 2413910

班级 软件四班

2025年5月15日

高级语言程序设计

实验报告

目录

[高级语言程序设计大作业实验报告 1](#_Toc1670257909)

[学生姓名 1](#_Toc118442979)

[1 作业题目 3](#_Toc2101290931)

[2 开发软件 3](#_Toc1028102402)

[3 课题要求 3](#_Toc663646652)

[4 主要流程 3](#_Toc2026701293)

[4.1 整体流程 3](#_Toc1530506384)

[4.2 类层次结构 4](#_Toc661672122)

[5 算法与机制 4](#_Toc1053030288)

[6 单元测试 5](#_Toc867315489)

[6.1 测试用例 5](#_Toc1999911434)

[6.2 测试结果 5](#_Toc97428394)

[7 开发遇到的问题与解决方案 7](#_Toc1096478944)

[7.1 角色控制 7](#_Toc964436901)

[6. 跳跃过快：连续跳跃。限制 isGrounded。 7](#_Toc84427551)

[7.2 角色技能 7](#_Toc1634642637)

[7.3 敌人 8](#_Toc680503988)

[7.4 UI 相关 8](#_Toc1880106041)

[7.5 动画 8](#_Toc867849129)

[3. 动作后卡住：未返回待机。设 Idle 为默认。 8](#_Toc231380679)

[7.6 代码问题 8](#_Toc1869670883)

[7.7 着色器与渲染 8](#_Toc1577807677)

[8 收获 9](#_Toc1085554183)

**1 作业题目**

本项目为基于 Unity 引擎开发的《Cuphead》风格 2D 动作游戏，采用 C#（模拟 C++ 面向对象编程）实现。玩家控制角色，支持左右移动、跳跃、下蹲、冲刺、普通攻击、 超级子弹和八方向锁定射击，与具有两阶段行为的 Boss 敌人对战。项目通过模块化设 计、状态机管理和物理优化，复现流畅的动画、精准控制和紧张的战斗体验。

**2 开发软件**

• **Unity** （版本 6000.0.43f1）： 2D 游戏引擎，负责渲染、物理、动画和场景管理。

• **Visual Studio Code**：用于编写和调试 C# 脚本，与 Unity 集成。

**3 课题要求**

1. **面向对象设计** ：通过类实现模块化组件，如玩家控制器、状态机、子弹系统。

2. **单元测试** ：验证移动、攻击、状态切换、碰撞、动画、音效和渲染。

3. **模型实现** ：开发玩家和敌人行为，清晰管理状态和交互。

4. **验证**：确保动画、物理、交互和渲染逻辑无缝运行。

**4 主要流程**

**4.1 整体流程**

项目采用面向对象设计，结合 Unity 组件化架构和状态机模式。主要组件：

• **玩家控制器**：管理移动（左右、跳跃、下蹲、冲刺）、攻击（普通子弹、超级子弹、 锁定射击）和状态切换（地面、下蹲、跳跃）。

• **敌人控制器**：实现两阶段 Boss，第一阶段跳跃追踪，第二阶段增强攻击。

• **动画系统**：通过 Animator 触发动画，与逻辑同步。

• **物理与碰撞**：使用 Rigidbody2D 和 LayerMask 优化交互。

• **渲染与特效**：使用 2D 精灵和粒子系统（如射击火花、跳跃烟雾）。

**4.2 类层次结构**

类结构如图 1所示：

• **GameObject**：Unity 基类，提供 Transform、Rigidbody2D。

• **PlayerController**：集成状态机、动画、音效，包含 FireBullet()、Dash()。

• **PlayerState**：抽象基类，定义 Enter()、Update()、Exit()。

• **PlayerGroundState 、PlayerCrouchState 、PlayerJumpState**：实现具体状 态。

• **Bullet**：管理子弹移动和伤害。

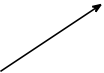
GameObject

Transform

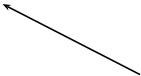


PlayerController FireBullet()

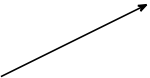
Dash()



Bullet damage



PlayerState Update()



PlayerCrouchState

PlayerJumpState

PlayerGroundState

图 1: 类层次结构图

**5 算法与机制**

1. **状态机控制**：PlayerStateMachine 切换地面、下蹲、跳跃状态，Update() 处理输入 （如“K”跳跃、“S”下蹲）。跳跃状态应用 jumpForce=15，触发 jumpSmokePrefab。

2. **攻击系统**：普通子弹（FireBullet()，速度 10，伤害 1），超级子弹（ShootSuperBullet()， 速度 15，伤害 100），锁定射击调整 firePoint。冷却分别为 0.2 秒和 5 秒。

3. **物理优化**：Rigidbody2D 处理移动（速度 8），Flip() 调整朝向，groundLayer 检 测 isGrounded。

4. **死亡与特效**：血量为 0 时触发 animator.SetTrigger("Die")，延迟 3 秒加载主 菜单。特效（如 fireDustPrefab）通过 Instantiate 管理。

5. **冲刺机制**：Dash() 协程实现冲刺（0.2 秒，3.5 单位），禁用重力，播放 dashClip。

**6 单元测试**

测试覆盖移动、攻击、状态、动画、音效、渲染和敌人交互。

**6.1 测试用例**

表 1: 单元测试用例

|  |  |  |
| --- | --- | --- |
| **输入** | **预期输出** | **测试目的** |
| 按右箭头 | 角色向右移动， 速度 8，动画 “Run” | 测试移动和动画 |
| 按“K”（地面） | 跳跃，力 15，播放 jumpClip，触 发烟雾 | 测试跳跃和音效 |
| 按“H” | 冲刺 3.5 单位， 0.2 秒，播放 dashClip | 测试冲刺 |
| 按“J” | 子弹朝向发射， 伤害 1，播放 shootClip | 测试普通攻击 |
| 按“Space”+“W”+ “J” | 子弹向上发射，偏移 (0,0.6)，动 画“AimUp” | 测试锁定射击 |
| 按“P”+ “L” | 超级子弹发射，伤害 100，播放 superClip | 测试超级子弹 |
| 按“P”，等 5 秒 | canUseSuperBullet 为真 | 测试冷却 |
| 按“S” | 下蹲，动画“Duck”，isDucking 为真 | 测试下蹲状态 |
| 血量为 0 | 死亡动画，播放 deathClip，3 秒 后加载主菜单 | 测试死亡逻辑 |
| 子弹击中敌人 | 敌人血量减 1 或 100 | 测试碰撞和伤害 |
| 运行粒子特效 | 射击火花、跳跃烟雾渲染，持续 0.2 秒和 1 秒 | 测试渲染 |

**6.2 测试结果**

初始测试发现：

• 子弹方向错误：修复 FireBullet() 的 sr.flipX 逻辑。

• 跳跃频率过快：添加 isGrounded 检测。

• 动画不同步：禁用 hasExitTime，调用 animator.SetTrigger。

• 超级子弹冷却失效：调整 ResetSuperBulletCooldown() 协程。

• 音效缺失：落地添加 landClip 播放。

• 粒子特效渲染失败：着色器编译错误（日志：Protocol error）。替换为 Sprites/Default 着色器，清理缓存。

所有测试用例通过，功能稳定。

**7 开发遇到的问题与解决方案**

**7.1 角色控制**

1. **角色重心不稳**：物理旋转影响控制。锁 y 轴，Flip() 调整朝向。

2. **向左移动不反转**：精灵未翻转。在 HandleMovement() 检查 moveInput，调用 Flip()。

3. **地面检测复杂**：检测不准。使用 groundCheckPoint 和 groundLayer。

4. **动画切换卡顿**：过渡停顿。禁用 hasExitTime，触发 SetTrigger("Aim")。

5. **落地无动画**：体验不完整。在 OnCollisionEnter2D 播放 landClip。

6. **跳跃过快**：连续跳跃。限制 isGrounded。

7. **技能时可移动**：不专业。禁用 moveInput，锁 rb.linearVelocity.x。

8. **锁定发射点错误**：未随方向调整。动态设置 firePoint.localPosition。

**7.2 角色技能**

1. **水花技能识别**：动画不完整。设 MeshType 为 Single。

2. **技能动画绑定**：未关联。检测按键，调用 SetTrigger("Aim")。

3. **子弹方向错误**：未随朝向。检查 facingRight，设 bullet.transform.localScale。

4. **超级子弹倒置**：精灵倒置。添加 SpriteRenderer.flipX。

5. **冷却状态不清**：无法判断。设 5 秒冷却，禁用 canUseSuperBullet。

6. **按键过快**：子弹连发。引入 canFire，间隔 0.2 秒。

**7.3 敌人**

1. **碰撞设置** ：物理碰撞受阻。取消碰撞，设触发器。

2. **攻击判定错误**：区域不准。延迟触发器激活。

3. **未攻击扣血** ：误伤。结合动画状态延迟判定。

4. **死亡动画秒放** ：未完整播放。延迟销毁，禁用碰撞体。

5. **不进入第二阶段** ：切换失败。修复血量逻辑。

6. **攻击弹飞自己**：位移异常。移除 Y 轴偏移。

7. **墓碑秒毁**：未抗击。添加血量脚本，设 500 血。

**7.4 UI 相关**

1. **血量** **UI 移动**：位置不固定。设 Screen Space - Overlay。

2. **UI 不显示**：未渲染。使用 Image 组件。

3. **受伤无反馈** ：UI 未更新。添加更新和闪烁效果。

**7.5 动画**

1. **子弹动画不播放**：未触发。调用 SetTrigger("Aim") 或"super"。

2. **动画切换停顿**：过渡延迟。添加过渡状态， ResetTrigger。

3. **动作后卡住**：未返回待机。设 Idle 为默认。

**7.6 代码问题**

• **定位困难**：错误难追踪。添加 Debug.Log()。

**7.7 着色器与渲染**

1. **着色器编译失败**：粒子特效（如 fireDustPrefab）未渲染，日志报“Protocol error

- failed to read magic number”。替换为 Sprites/Default 着色器，清理缓存，重 启 Unity。

**8 收获**

在做项目之前， 我其实对软件工程中的一些规范——比如命名法、文档分类、职责分离 这些东西理解很浅， 甚至觉得没那么重要。平时写作业时， 只要能跑就行，变量名可能 随便叫 x1 、temp 、aabbcc，函数写在哪都可以，文档更是不写。但在这个项目实际开 发中，我遇到的第一个问题就是——代码一多，就看不懂自己写了啥。

比如我用了 isDuck 、DuckNow 、duckIdle 、ducking 等变量，后来想判断角色到底是不 是下蹲，根本分不清谁是谁。 一段时间后，我发现我写过一段冲刺逻辑， 但是完全找不 到函数在哪——因为我没有统一命名，也没有注释。

所以后来我重构了大部分命名，统一用驼峰命名法（CamelCase）， 并按变量功能分类， 比如：状态布尔变量以 is 开头，如 isDucking、isGrounded；控制变量带有清晰限定， 如 canFire 、canUseSuperBullet。

同时，我也开始写注释、拆分逻辑到不同方法。现在回头看， 整个代码逻辑清晰了很多， 调试效率也更高了。软件工程的很多“形式主义”，只有你亲自做过一个完整的项目，才 会真正明白它的意义。它不仅是制定一个通俗的规则方便团队协作， 更是为了你自己后 续能理解、维护和扩展你的代码。