華中科技大學

《软件工程》项目报告

题目：吃豆人（Pac-Man）

课程名称：软件工程

专业班级：CS1802

学　　号：U201814531

姓　　名：李响

同组成员：黎奕辰

柳昕

指导教师：刘宏

报告日期：2020.12.27

计算机科学与技术学院

任 务 书

一 总体要求

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；
2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。
3. 根据需求分析进行总体设计、详细设计、编码与测试等。

二 基本内容

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

1. **问题概述、需求分析**：正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如 NABCD 模型，Microsoft Visio，StarUML 等工具 (20%)；
2. **原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；
3. **编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；
4. **功能创新**：与众不同、特别吸引用户的创新（10%）；
5. **用户反馈**：包括用户的使用记录，照片，视频等（5%）。

目　录

1 问题定义

1.1 项目背景与意义

在进行项目的前期调研时，本小组成员对项目的选择进行了相应的讨论，我们最终选择了在现代 PC 主机（Windows 系统）上复现吃豆人（Pac-man）游戏作为我们的项目目标。《吃豆人》（Pac-Man）是一个经典的游戏，最早由南梦宫公司于 1980 年在街机上推出，曾经这款街机游戏具有广泛的用户群体，是许多人的童年的重要记忆。本次复现吃豆人游戏，不仅要使其能够在现代 PC 计算机上运行，同时还要对游戏内容进行扩充和创新，具体项目分析如下：

1.1.1 需求 Need

《吃豆人》（Pac-Man）是一个经典的游戏，最早由南梦宫公司于 1980 年在街机上推出。作为一款经典的街机游戏具有广泛的受众，现在仍然有许多想要玩吃豆人（Pac-Man）游戏的用户，但是，这些游戏用户常因为游戏版本和机器不适配或者可以运行游戏的机器过于老旧而让放弃玩吃豆人游戏，用户希望有一款可以在现代 PC 主机上运行的，功能完善、运行流畅的吃豆人游戏。

1.1.2 做法 Approach

使用 Unity 集成开发平台进行《吃豆人》（Pac-Man）游戏的开发，游戏开发语言为 C#编程语言，图片资源和音效资源使用网络公开的免费数据资源。使用 Unity 平台，实现一款可以在 PC 主机上运行的吃豆人游戏项目目标，同时，在原有游戏的基础上进行游戏功能的创新和修改，以求给用户更完美的体验。

1.1.3 益处 Benefit

《吃豆人》是一款可以带给人愉快感和畅快体验的游戏，作为一款经典街机游戏，是广大青年人共同的童年记忆，吃豆人虽然相较于现代大型复杂的游戏游戏架构和设计都略显简陋，但是仍然可以给整日匆匆的人提供一段忘记烦恼、体验快乐的时间，让那些曾经通过这款游戏连接在一起的同龄人回忆儿时的快乐，暂时放下当下的烦恼。

1.1.4 竞争 Competitors

目前网络上各种游戏成千上万，种类繁多，《吃豆人》作为其中的一个更是个经典的游戏，现有的各个版本的《吃豆人》游戏也很多，因此我们的游戏项目存在较大的竞争压力。

但是，我们计划在原有的吃豆人基础上对游戏进行优化和创新，力求在保留原有游戏风格的前提下开发更多功能，增加新的游戏道具。通过优化和创新的吃豆人相比于同类游戏有着一些优势。

1.1.5 推广 Delivery

可以在自己的朋友圈和 QQ 空间等地方进行推广，让朋友试玩并帮忙扩散；在学院年级群进行宣传和推广，让同学们试玩我们的游戏并帮助推广；联系自媒体公众号进行推广（需要支付一定的广告费用）。

1.2 项目基本目标

在原吃豆人（Pac-man）游戏的基础上，在现代个人计算机（Windows 系统）上对其功能进行复现，实现在 PC 机上运行吃豆人游戏的目标。并在原游戏的基础上，增加原游戏没有的新道具和新地图，以期在保留吃豆人游戏原有风格的前提下，给用户全新的体验。在项目实现过程中，我们将使用 Unity 进行游戏开发平台，码云进行代码管理。具体而言，该游戏应该具有如下功能：

1. 玩家在进入游戏后应该可以选择开始游戏或者退出游戏，此外还需要添加关闭音效等附加功能按钮。

2. 进入游戏后，玩家可以使用键盘的上下左右按键对吃豆人进行实时控制。

3. 吃豆人获胜的标准为吃光所有的豆子，且在游戏过程中不遇到鬼；若遇到鬼，游戏失败，返回初始界面。

4. 游戏会定时随机在地图中生成道具，道具具有特殊效果。

1.3 可行性分析

从技术角度对项目目标进行分析，由于本项目的目标是在复现原有的吃豆人游戏基本功能的基础上增加新的道具和场景，项目的目标非常清晰，各种图形资

源都可以在网上比较方便地获得，怪物和吃豆人的移动等在技术上的实现也较为简单。此外，本项目计划使用 Unity 在 PC 机上设计并实现吃豆人游戏，Unity 提供了一整套软件解决方案，适合在较短时间内开发出界面友好、美观，内容丰富的游戏；Unity 的开发语言 C#类似与 C++，比较好上手。

从市场需求的角度来分析，市场上存在数量较多的较为怀旧的用户，我们的项目在解决这类用户的需求方面有一定的市场，在对吃豆人游戏进行优化和创新之后，因此我们的项目有较为广阔的市场前景。

在经济上，由于项目的开发可以在其他活动的空闲时间完成，且不需要设立专门的开发设备和地点，因此不需要额外经济成本，在经济上也有较强的可行性。

1.4 人员管理和项目进度管理

1.4.1 人员管理

本小组的成员共 3 人，分别为柳昕、李响和黎奕辰，由于本次项目的工程量较小，且参与人数较少，因此小组采用民主制组织模式进行交流，不设立专门的组长进行任务分配，小组具体分工如下：

1. 柳昕：

本次项目中，主要完成了游戏中图片资源和声音资源的收集，辅助黎奕辰同学完成一些模块中函数的设计。在游戏项目基本完成以后提出相关的建议。在报告中，撰写了主要函数声明，关键函数说明，测试计划和测试用例，结果分析，并且收集部分用户使用以后的反馈信息（**3.3 到 4.4**）。

2. 李响：

主要负责前期的项目分析和模块划分，主要参与了 NABCD 模型构建、可行性分析、以及项目整体系统架构设计和对象模型设计等总体性工作，在本次报告撰写中，主要编撰了报告中从 1 问题定义到 3.2 对象模型分析的内容（**1 到 3.2**）。

3. 黎奕辰：

本次项目中我的工作主要是负责游戏模块和代码的具体设计，编写游戏脚本，合并组员传输的模块，并使用柳昕同学提供的声音，图片等资源在 Unity 上构建出完整的游戏雏形。实验报告主要负责 3.4，3.5 的部分（**3.4&3.5**）。

1.4.2 项目进度管理

系统结构如图 1.1 所示，项目根据输入输出可以划分为 3 个主要模块，用户操作接收模块、资源状态修改模块以及场景显示模块，这三个模块由可以划分成 9 个子模块，具体进度计划安排如下：

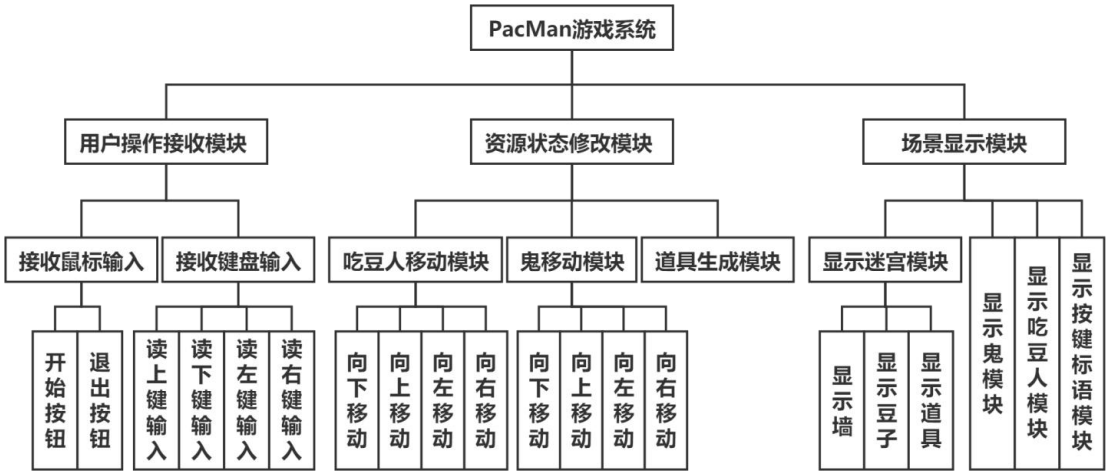

图 1.1 吃豆人游戏系统结构图

1. 用户操作接收模块，编码需要 1 天，测试 1 天，调试 1 天；
2. 资源状态修改模块，编码需要 2 天，测试 1 天，调试需要 1 天；
3. 场景显示模块，编码需要 4 天，测试 1 天，调试 2 天；
4. 最后汇总所有模块后，集成测试需要 3 天完成。

2 需求分析

2.1 吃豆人 E-R 图

对经典的吃豆人游戏进行分析后，我们对吃豆人游戏进行了抽象，得到了如图 2.1 所示的 E-R 数据模型图。吃豆人游戏主要包括**迷宫**、**鬼**以及**吃豆人** 3 类主要实体，用户作为系统外的实体，其年龄等个人属性对于此系统的意义不大，故图中只显示了用户偏好这一属性。

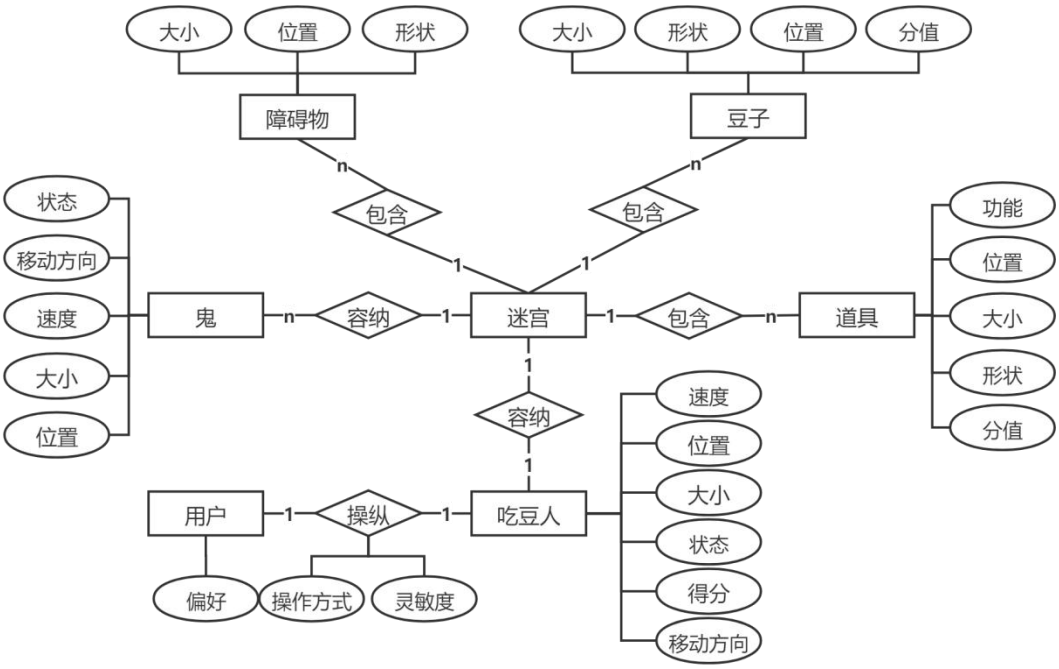

图 2.1 吃豆人游戏 E-R 图

迷宫实体中包含若干其他实体，如**障碍物**实体、**豆子**实体以及**道具**实体等；此外，鬼和吃豆人都在迷宫中移动，吃豆人个数一定为 1，而鬼的个数可以由设计者设定，通常为 4 个，不宜过多。

迷宫中各种实体（**障碍物**实体、**豆子**实体以及**道具**实体）有各自的属性，其中形状、大小以及位置为其公有属性，豆子和道具额外具有分值属性，道具还具有产生功能的属性。鬼和吃豆人作为可移动实体，在基本形状、大小以及位置属性外，还应包括移动的方向与速度等，具体如上图所示。

用户其个人属性对于游戏系统无意义，因此仅将用户偏好作为其属性，用户操纵吃豆人进行游戏，其操作纵过程具有属性操作方式以及灵敏度等，可以通过调节这些值给与用户更好的体验。

2.2 吃豆人数据流图

使用数据流图对系统的基本功能进行分析，吃豆人作为一款经典游戏，系统的源点和终点都是用户，用户通过某个操作设备对游戏进行操作，游戏系统将结果以图像和音效的方式反馈给用户，可得顶层 DFD 图如图 2.2.1 所示。

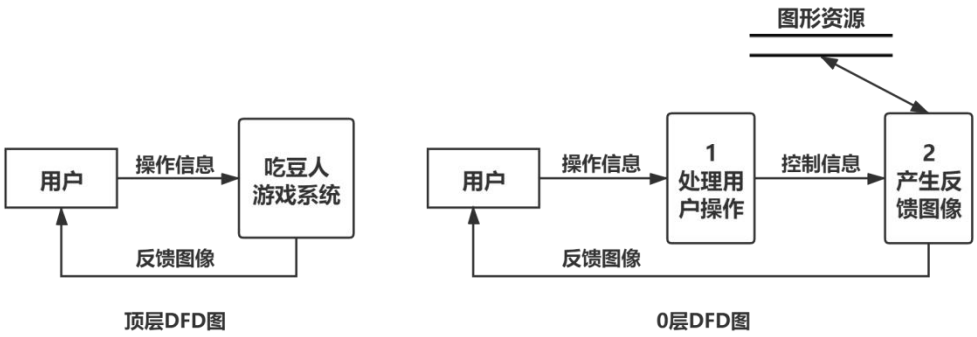

图 2.2.1 吃豆人游戏顶层与 0 层数据流图

对于游戏系统进行细化，系统需要分为**处理用户操作信息**和使用操作信息**改变资源并显示反馈信息**两个部分，可得到 0 层 DFD 图，如图 2.2.1 所示。然后，对两个系统进一步细化，就可以得到如图 2.2.2 所示的最终数据流图。

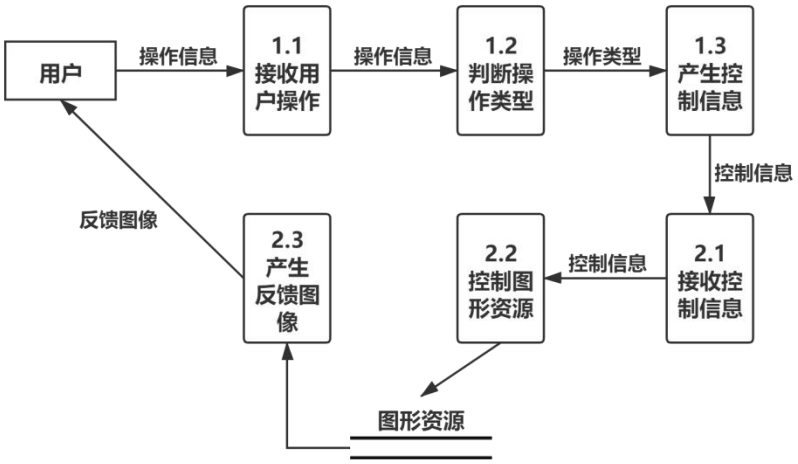

图 2.2.2 吃豆人游戏最终数据流图

2.3 吃豆人状态转换图

使用状态转换图对系统的状态及引起系统状态转换的事件进行分析，主要有 4 个主要状态，初始状态、游戏开始状态、游戏进行状态及游戏结束状态。

其中，初始状态为未打开游戏时的状态，为未开始状态；游戏进行状态为游戏正在进行时的状态，具体化到本项目为可以操纵吃豆人时的状态，此状态可以进一步进行划分，当遇到障碍物、豆子、道具以及鬼的时候都会对当前状态产生

改变和转换，如遇到障碍物禁止前进，遇到道具会产生特别效果等等；游戏结束状态可以包括失败和获胜，分别由遇到鬼和吃光豆子触发。通过以上分析，可得到如图 2.3 所示状态转换图。

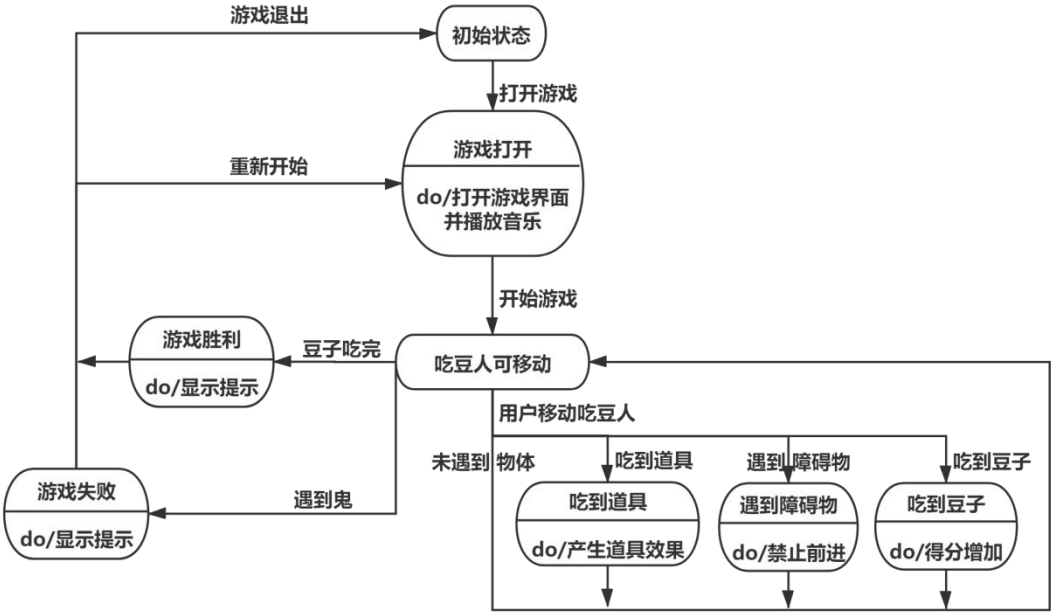

图 2.3 吃豆人游戏状态转换图

2.4 吃豆人游戏用例图

使用用例图对用户的需求和操作进行分析，明确其目标。通过分析可知，吃豆人游戏需要为用户提供 4 个必须功能以及 1 个附加功能，必须功能包括打开游戏功能、开始游戏功能、退出游戏功能与操控吃豆人功能，附加功能为关闭/打开音效功能（此功能主要为提高用户体验设计），用例图如图 2.4 所示。

其中，操控吃豆人包含向上、下、左、右移动吃豆人的基本功能。

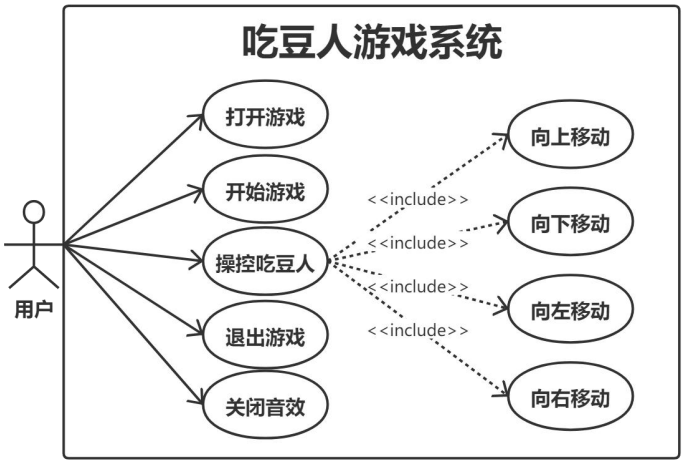

图 2.4 吃豆人游戏用例图

2.5 原型系统设计

本次项目基于 Unity 进行开发，本项目主要分为 UI 界面模块和资源操作逻辑模块，资源操作逻辑模块主要负责游戏资源的控制，该模块需要对玩家的操作进行响应并对资源进行适当、正确的处理，UI 界面模块则主要提供用户界面和操作接口，方便用户的游戏操作。

本次项目主要运用网上公开的图片素材资源进行界面的设计。本项目主要涉及 4 个方面的主要素材运用，第一方面是迷宫素材的建造和使用，包括障碍物、豆子以及道具等素材的设计；第二方面是吃豆人素材的使用，包括其移动方式以及和道具、鬼等其他素材的相互作用；第三方面是鬼素材的使用，包括其移动方式以及和道具、吃豆人等素材的互相作用；第四方面为其他界面素材（如开始、结束界面），以及音效等素材。

通过出初步的分析和设计，我们设计了如下的原型系统，通过对素材的简单布置和堆叠可以看到系统的大致雏形，具体成果如下：

1. 使用 Unity 生成功能实现可执行程序 PacMan.exe，玩家只需要点击游戏即可运行该游戏，玩家点击“PacMan.exe”打开游戏，可以看到如图 2.5 所示的游戏开始画面，可以看到两个选项“start”和“exit”，可以开始游戏或退出游戏，若点击 start 则进入游戏开始界面，若点击 exit 则游戏关闭。

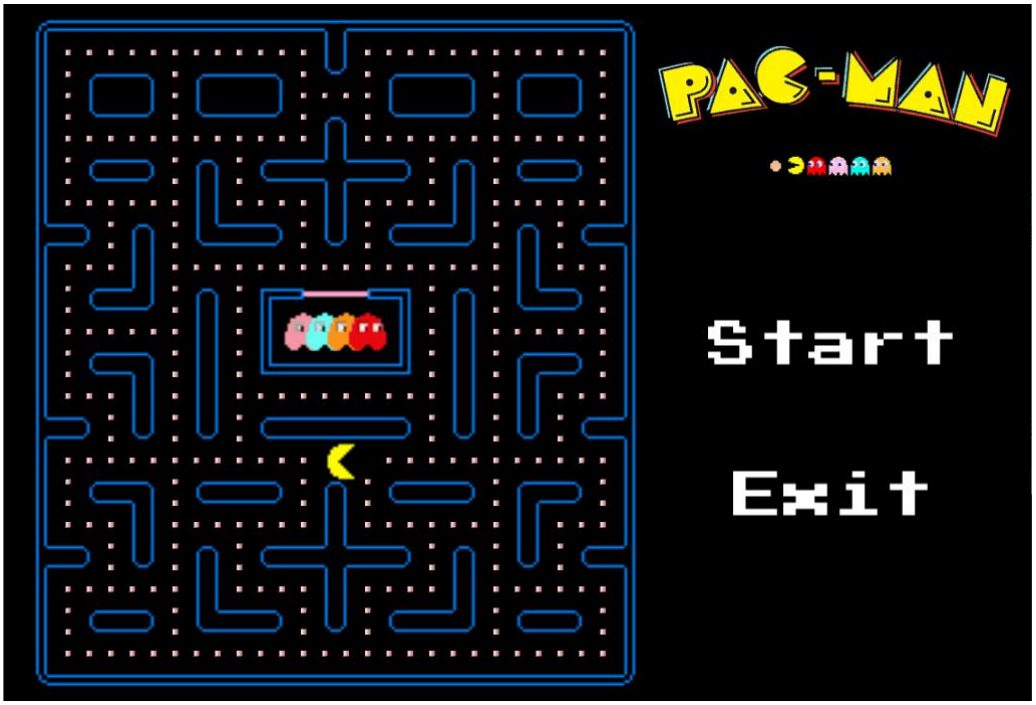

图 2.5 吃豆人游戏开始界面截图

2. 游戏启动后，玩家可使用键盘操作吃豆人移动，玩家需要在躲避鬼的基础上吃完所有的豆子，方能获得游戏胜利，吃豆人通过方向键进行操纵，具体的操作方法如下：

（1）方向键↑：吃豆人向上方转向并向上方前进

（2）方向键↓：吃豆人向下方转向并向下方前进

（3）方向键←：吃豆人向左方转向并向左方前进

（4）方向键→：吃豆人向右方转向并向右方前进

3. 本游戏中使用了部分网上公开的资源，主要图片资源如图 2.6 所示，下面对各种资源进行详细描述：

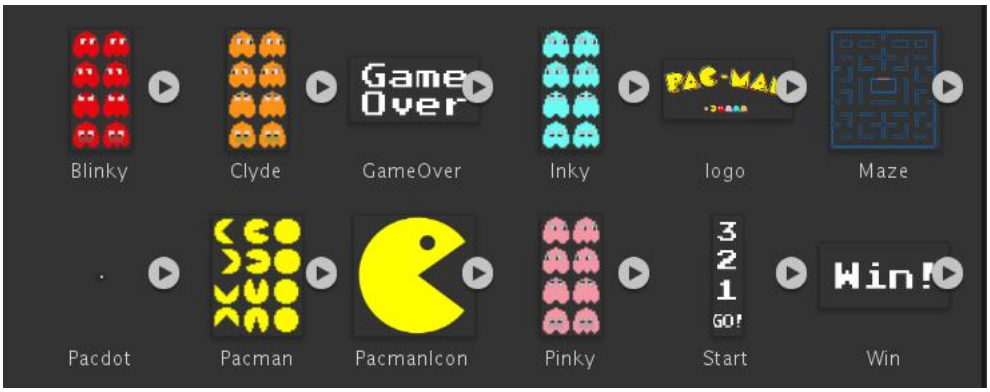

图 2.6 吃豆人游戏图片资源截图

（1）鬼的图片资源有 4 种，分别为 Blinky、Clyde、Inky 以及 Pinky，代表不同的鬼，使用 Unity 对图片进行分割可以将鬼的不同状态进行分割，配合动作器，可以实现转向时的资源状态的切换；以 Blinky 为例如图 2.7 所示，将其切分成 8 个部分，配合动作器 Animator 进行控制即可实现此功能。

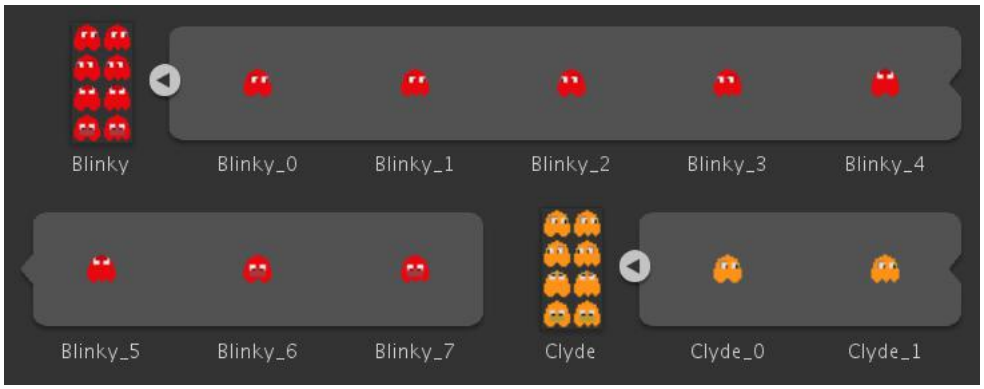

图 2.7 吃豆人游戏 Blinky 图片资源切分示意图

（2）与鬼资源的切分与控制相似，吃豆人图片资源也需要使用 Unity 对吃

豆人图片资源进行分割，配合动作器，可以实现转向时的资源状态的切换；如图 2.8 所示，将其切分成 12 个部分，配合动作器 Animator 进行控制即可实现吃豆人的逐帧动画功能。

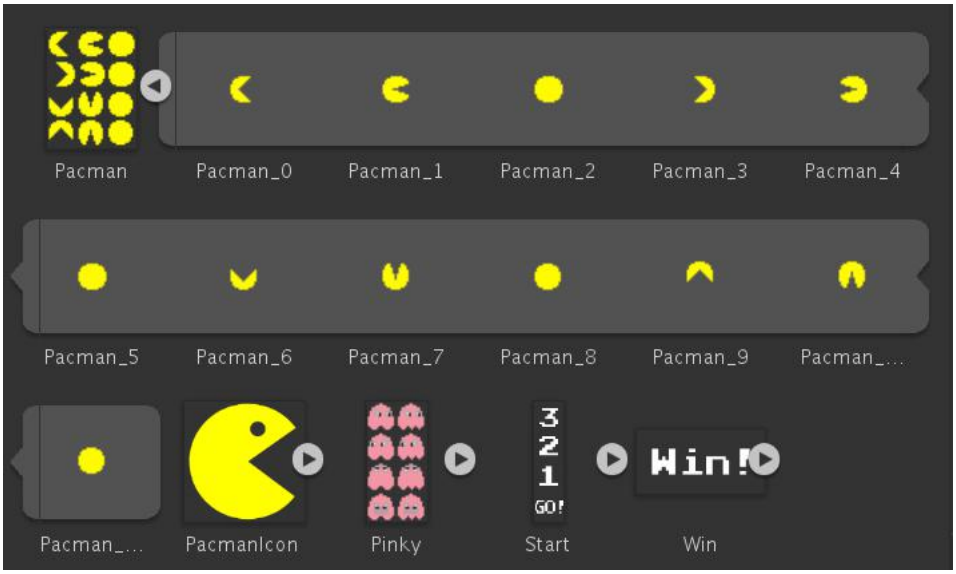

图 2.8 吃豆人游戏 Pacman 图片资源切分示意图

（3）使用 Unity 的组件刚体碰撞器 Box Collider2D 来设计迷宫中的障碍物墙，图片资源豆子填充在路径上，如图 2.9 所示。

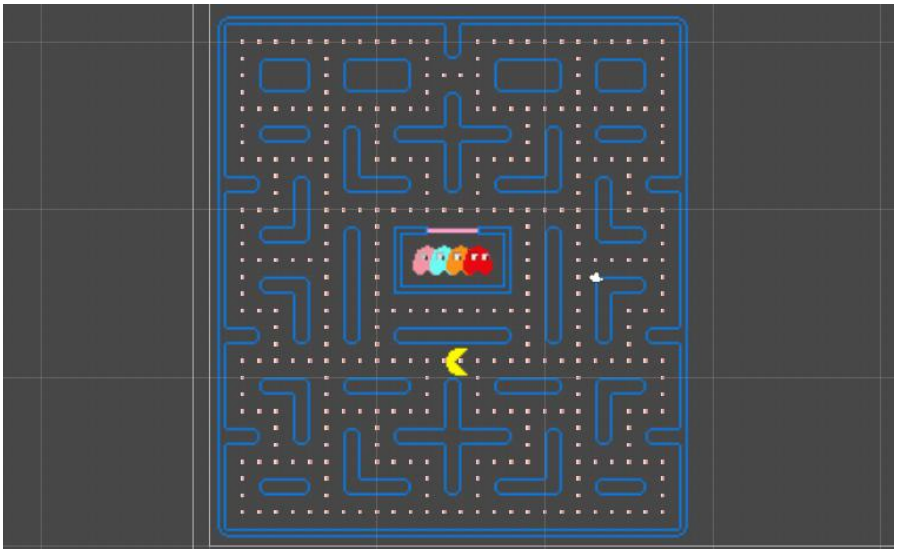

图 2.9 吃豆人游戏迷宫示意图

4. 进入游戏界面后，如图 2.10 所示，右侧的开始按钮和退出按钮变成了提示标语，分别提示当前剩余的豆子数和已经吃掉的豆子数，最下一行 score 表示当前的得分数，每个普通豆子的分数为 100，道具豆子的分数为 200，在鬼被冻结期间遇到鬼可以额外获得 500 分。

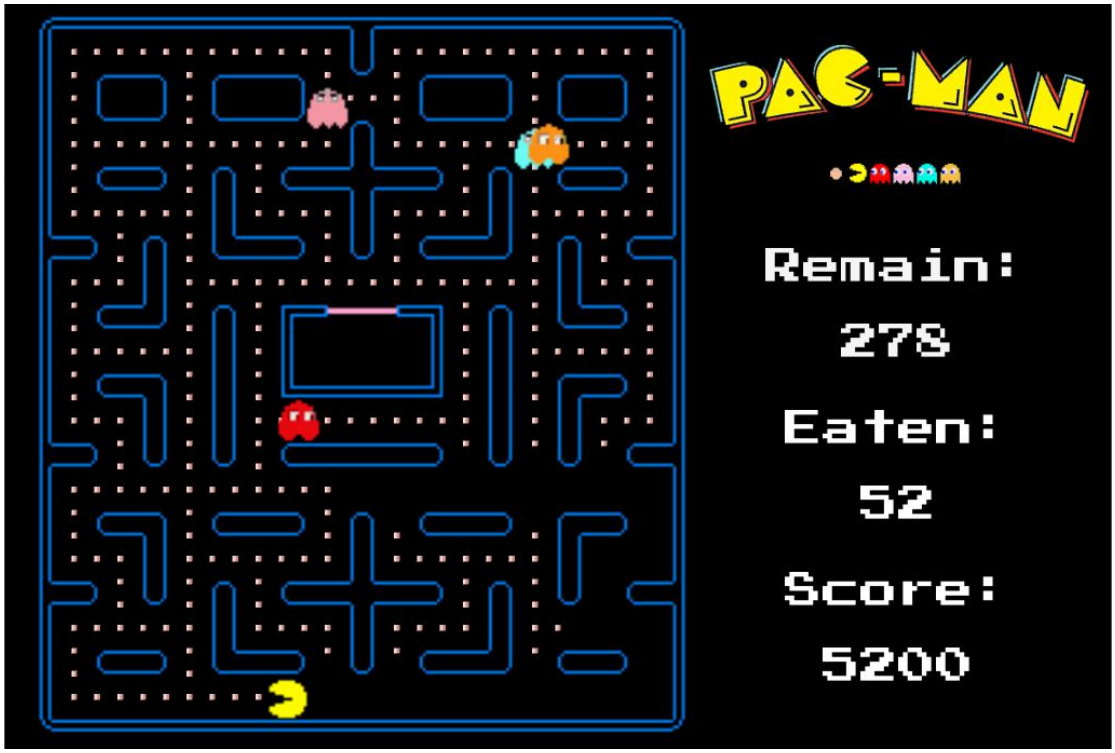

图 2.10 吃豆人游戏正在进行画面截图

5. 当吃豆人在普通状态下，遇到鬼会导致游戏失败，如图 2.11 所示，游戏界面会显示提示标语“Game over”，表示游戏结束；如果吃豆人吃光了所有的豆子，如图 2.12 所示，游戏界面会显示提示标语“Win!”，表示游戏胜利。

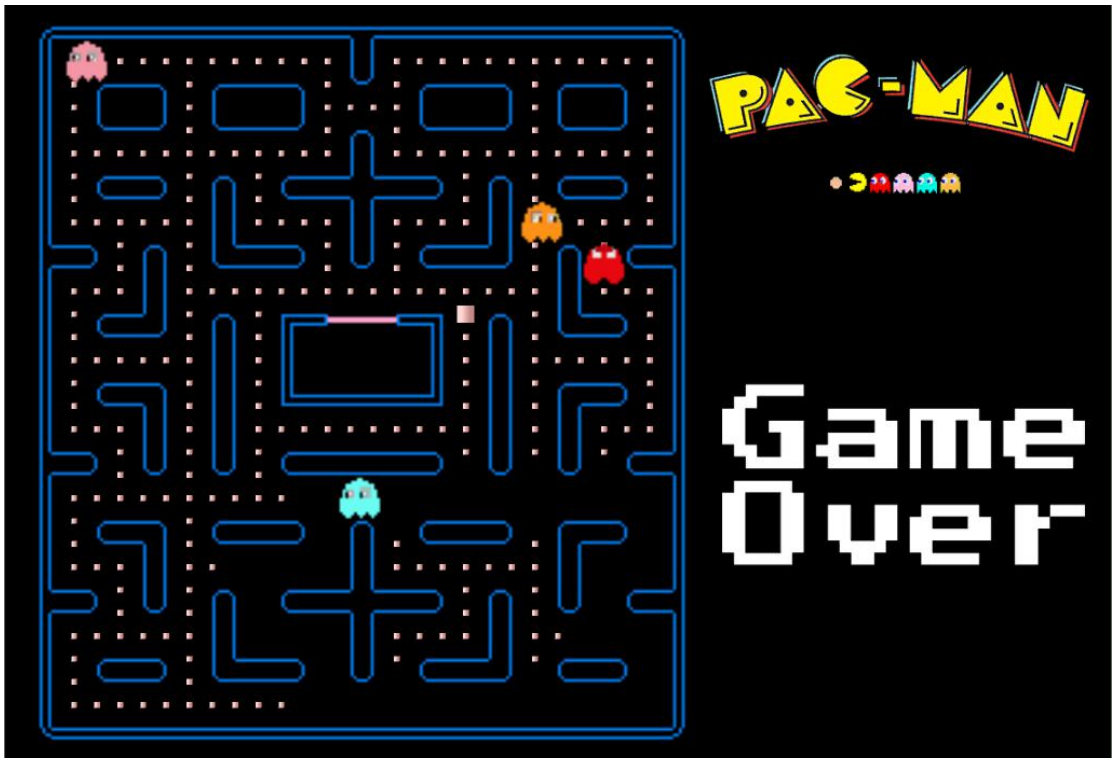

图 2.11 吃豆人游戏游戏失败画面截图

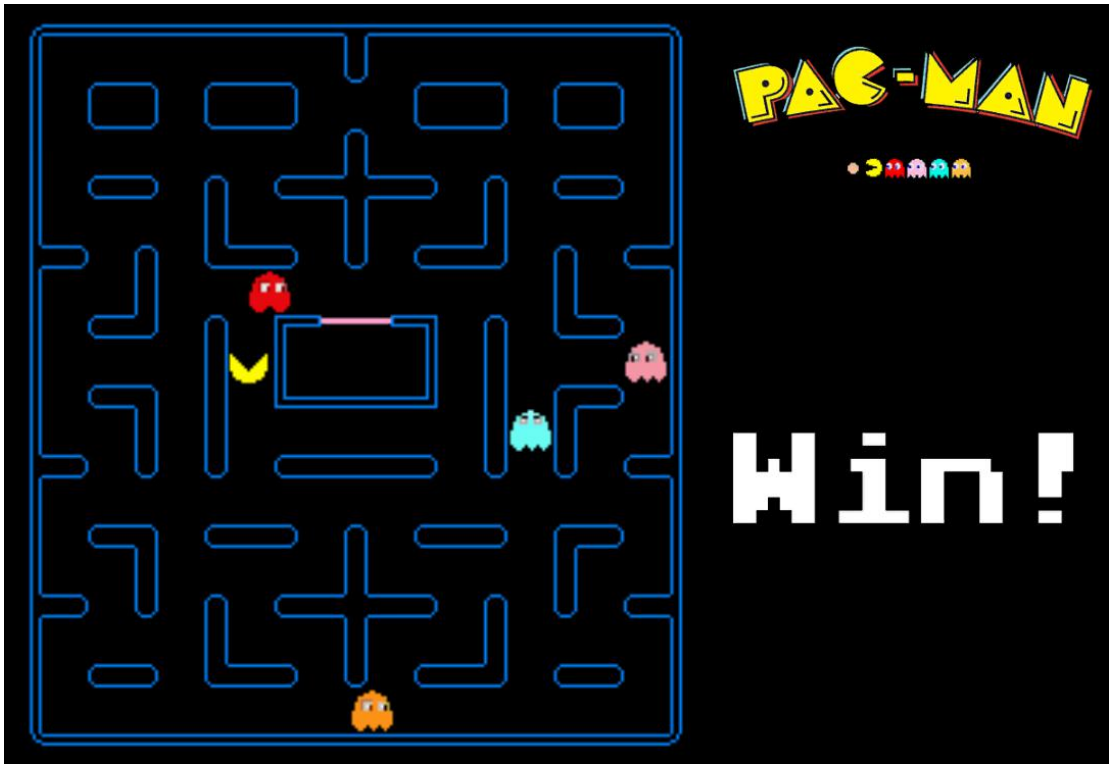

图 2.12 吃豆人游戏游戏胜利画面截图

6. 当吃豆人吃到超级豆子道具时，吃豆人进入超级豆子状态，4 个鬼会被暂时性冻结 3 秒钟时间，在此期间如图 2.13 所示，鬼会停止运动且颜色发生变化，当吃豆人在此期间遇到鬼可以将鬼重置回出发点，并获得额外的得分。

图 2.13 吃豆人游戏超级豆子道具生效画面截图

3 概要设计和详细设计

3.1 系统结构设计

采用面向数据流的设计方法分析吃豆人游戏系统的系统结构，精化系统数据流图后对数据流进行分析，可将整个游戏系统分为用户操作接收模块（输入 I）、资源状态修改模块（操作 P）以及场景显示模块（输出 O）。

其中，用户操作接收模块的主要功能为检测与接收用户操作，用户主要通过**键盘操作**与**鼠标操作**来完成游戏操作；资源状态修改模块的主要功能为修改各个图片资源的状态，包括图片种类、位置、移动速度以及方向等；场景显示模块的主要功能为整合资源布局并显示场景。系统结构图如图 3.1 所示，各模块模块划分如下：

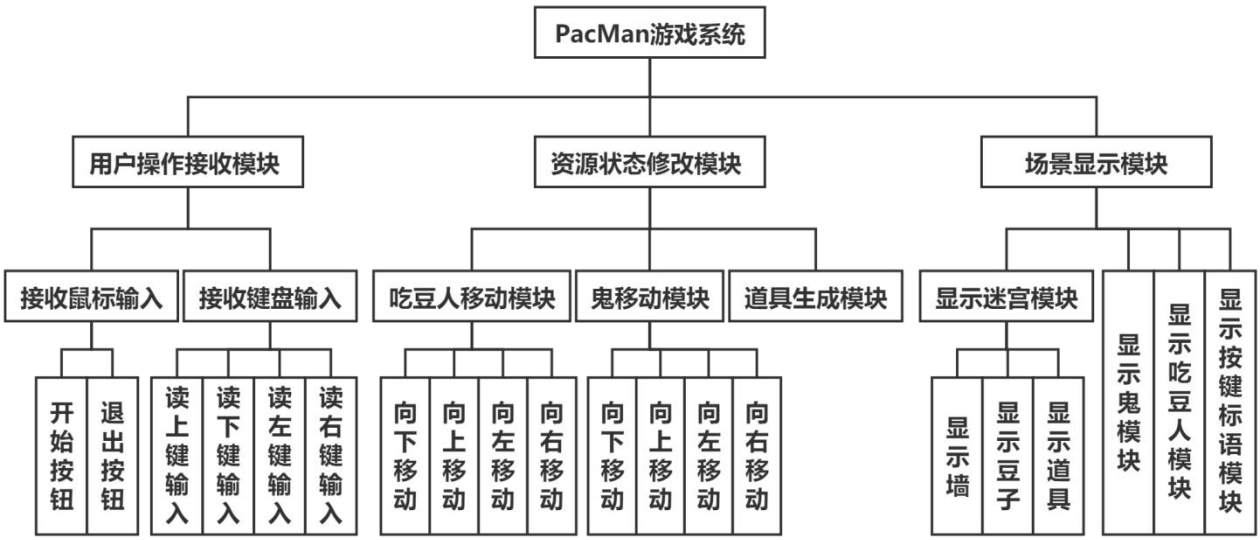

图 3.1 吃豆人游戏系统结构图

3.1.1 用户操作接收模块

用户操作接收模块由鼠标输入接收模块和键盘输入接收模块组成，具体模块划分和接口描述如下：

（1）鼠标输入接收模块，该模块的主要功能为检测并处理鼠标点击按钮时产生的事件，主要需要检测开始按钮以及退出按钮的触发情况，本模块主要使用 Unity 自带的 Input 模块进行操作，使用 Unity 自带的时间处理函数即可。

（2）键盘输入接收模块，该模块的主要功能为检测并处理键盘输入产生的

事件，需要检测上、下、左、右四个按键的触发情况，本模块主要使用 Unity 自带的 Input 模块进行操作，使用 Unity 自带的时间处理函数即可。

3.1.2 资源状态修改模块

资源状态修改模块由吃豆人移动模块、鬼移动模块和道具生成模块组成，具体模块划分和接口描述如下：

（1）吃豆人移动模块，本模块为控制吃豆人移动的脚本模块。本模块通过键盘输入接收模块的接口，得到吃豆人的移动方向和时间信息，以此更新吃豆人实例资源的状态情况；同时，将状态信息传递给吃豆人显示模块的接口，使其能够正常显示。

（2）鬼移动模块，本模块为控制鬼移动的脚本模块。本模块控制鬼进行随机移动，并状态信息传递给鬼显示模块的接口，使其能够正常显示。

（3）道具生成模块，本模块控制道具的生成，每隔一段时间会随机将一个豆子转换成道具，道具拥有特殊的功能，当吃豆人吃到道具的时候会进入特殊状态，具有特殊的效果。

3.1.3 场景显示模块

场景显示模块由显示迷宫模块、显示鬼模块、显示吃豆人模块和显示按钮标语模块组成，具体模块划分和接口描述如下：

（1）显示迷宫模块，本模块的主要功能为显示迷宫资源，其中包括障碍物（墙）、豆子以及道具等，其中障碍物使用 Unity 的刚体实现，豆子使用图片资源进行放置，道具由豆子转化而成，具有特殊的功能，需要提供特殊状态接口，当其遇到吃豆人时，吃豆人需要进入特殊状态。

（2）显示鬼模块，本模块的主要功能为显示鬼，鬼的显示需要根据鬼的运动方向显示不同的图片资源，且要根据位置等信息显示其图片，因此本模块向其他模块提供了两个主要接口，一个是方向接口，一个是位置接口。

（3）显示吃豆人模块，本模块的主要功能为显示吃豆人，其显示原理和提供接口与显示鬼模型类似，不再赘述。

（4）显示按钮标语模块，本模块提供标题、开始按钮、结束按钮以及一些提示信息的显示功能，本模块需要调用鼠标点击信息检测模块进行检测，还需向

其他模块提供接口以显示信息。

3.2 对象模型设计

本次项目的编程语言为面向对象的语言C#，使用的开发平台为Unity，因此采用Unity的类组织模式进行设计与开发，Unity使用组件来封装功能，由于使用了部分Unity系统自带的类，其属性和方法过多，因此部分类图的属性和行为不完全写明，仅将类名写明。吃豆人游戏系统类图如图3.2所示，具体的类的封装和设计如下：

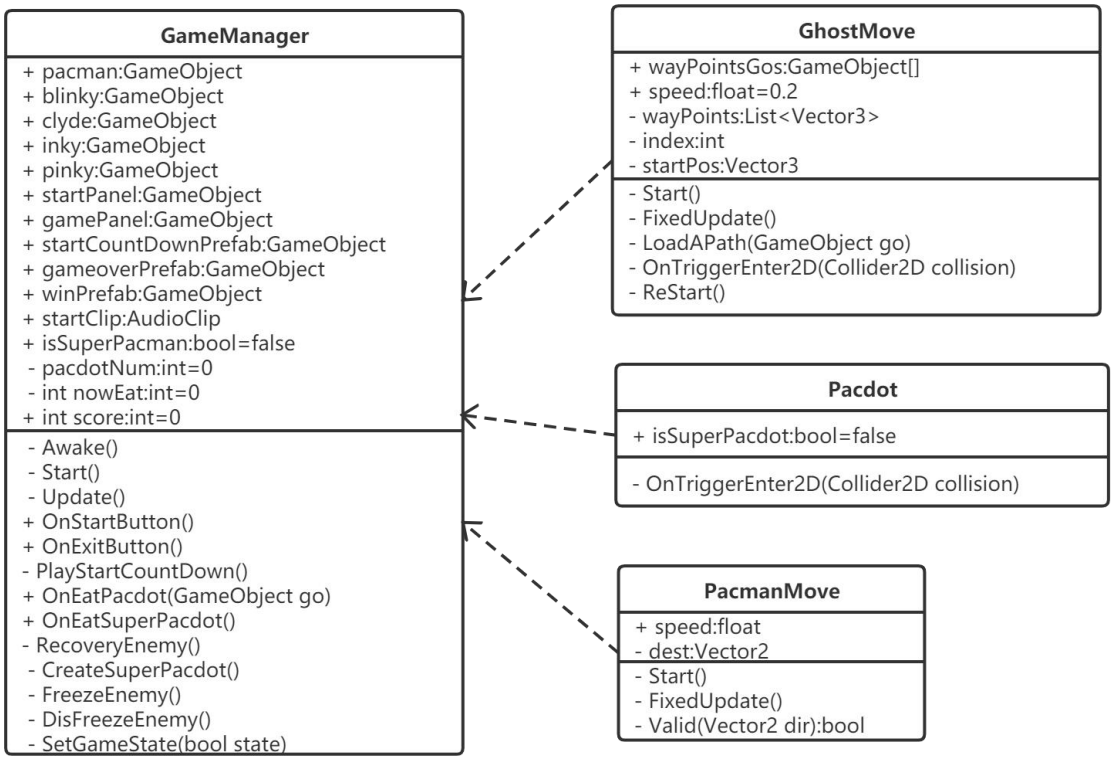

图 3.2 吃豆人游戏类图

1. GameManager类

GameManager类的主要功能为向其他类提供接口和方法，作为管理游戏操作的管理器使用，其中前11个数据成员为GameObject类成员需要与相对应的游戏实例相绑定，isSuperPacman为布尔型成员，其辨识吃豆人的状态；pacdotNum和nowEat分别为豆子的数目和吃掉的豆子的个数；score为当前分数；各个方法的主要功能如下：

（1）Awake()以及Start()，负责界面的初始化；

（2）Update()，该方法会被周期性调用，以达到刷新的目的；

（3）OnStartButton()，该方法实现按下开始按钮后的操作；

（4）OnExitButton()，该方法实现按下退出按钮后的操作；

（5）PlayStartCountDown()，该方法协助OnStartButton()实现标语的显示；

（6）OnEatPacdot(GameObject go)，该方法实现吃到豆子后的操作；

（7）OnEatSuperPacdot()，该方法实现吃到超级豆子后的操作；

（8）CreateSuperPacdot()，该方法创建超级豆子；

（9）FreezeEnemy()，该方法冻结敌人；

（10）DisFreezeEnemy()，该方法取消冻结敌人；

（11）SetGameState(bool state)，该方法设置游戏初始状态；

2. GhostMove类

该脚本控制鬼的移动，使用A*算法实现鬼的自动寻路功能，当游戏开始后，鬼会使用寻路算法找到一条路径进行移动，当鬼和吃豆人碰撞时，若吃豆人属于普通状态则游戏结束，若吃豆人属于超级状态则退回原处且得分增加，具体函数的功能如下所示：

（1）Start()，该方法实现对鬼的初始化；

（2）FixedUpdate()，该方法周期性调用实现资源的刷新；

（3）LoadAPath(GameObject go)，该方法会加载一条路；

（4）OnTriggerEnter2D(Collider2D collision)，该方法实现碰撞实现；

（5）ReStart()，该方法实现失败后的游戏重启；

3. Pacdot类

该脚本只有方法OnTriggerEnter2D(Collider2D collision)，其作用为控制豆子碰撞实现，若吃豆人吃到普通豆子，得分增加豆子销毁；若吃豆人吃到超级豆子，得分增加、进入超级状态并销毁超级豆子，具体函数的功能如下所示：

4. PacmanMove类

该脚本控制吃豆人的移动，当游戏开始后，用户通过键盘控制吃豆人的移动，具体函数的功能如下所示：

（1）Start()，该方法实现对吃豆人的初始化；

（2）FixedUpdate()，该方法周期性调用实现资源的刷新；

（3）Valid(Vector2 dir)，该方法检测将要去的位置是否可以到达；

3.3 关键数据结构定义

由于是面向对象编程，这些数据都作为类的成员，单独存放在类中，相互之间没有明显的耦合关系，具体的数据结构的定义如下所示：

1. 音频数据，主要包括开场音乐和背景音乐，音频数据可以利用Unity自带的AudioClip和AudioSource数据类型进行定义与存储；

2. 图片数据，主要包括豆子、超级豆道具、吃豆人以及鬼等基本图片资源，以及音量的开启和关闭图片等其他附加的资源数据，这些数据可以使用Unity自带的Sprite数据类型进行定义与存储；

3. 预制体数据，地图边沿和墙的实体，在初始化迷宫时进行使用，这些实体利用Unity自带的Unity数据类型进行定义与存储；

4. 地图信息，这些包含吃豆人，鬼和豆子的位置信息，每个位置都可以用一个Vector3存储，然后这些Vector3汇合成一个list。

3.4 关键算法设计

吃豆人游戏系统总共分为四个模块，分别为游戏管理GameManager模块、鬼魂移动模块、吃豆人移动模块以及豆子模块，各个模块具体算法设计如下：

3.4.1 游戏管理模块设计

本模块主要用来实现对游戏的整体控制和局部的一些细节方式实现，这个部分包含三个主要函数，第一个是初始化唤醒函数awake函数，第二个是开始函数start函数，第三个是更新函数Update函数，具体实现如下。

第一，awake函数的实现过程，在游戏开始的时候，首先，使用screen类的函数设置屏幕大小；随后，使用random类的函数生成迷宫和鬼的诞生点；再通过find函数，找到迷宫每一个可以通过的结构物品添加豆子；最后，把豆子的数量设置成预置的值即可实现初始化功能。

第二，start函数的实现过程，它会设置好声音按钮的功能，然后调用库函数SetGameState让所有生物移动状态停止。

第三，Update函数的实现过程，函数流程图如图3.3所示，首先，该函数需要判断胜利的条件，即豆子已经被吃完而且吃豆人没有碰到怪物，如果胜利，则

显示胜利提示并重置游戏，豆子吃完时可以使用anykeydown函数使用户输入任意键时重新开始游戏，同时，需要实时更新游戏的分数，豆子数这样时刻可能会变的量。其他的函数都是处理一些细节地方功能，不能算关键函数，故不在此详细讲解。

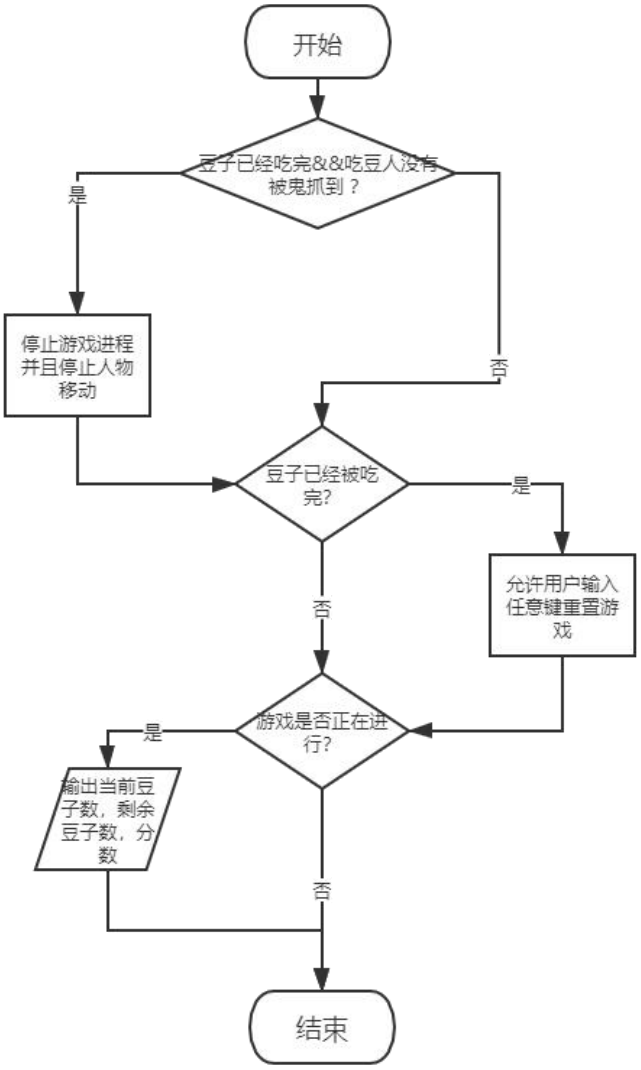

图 3.3 Update 函数流程图

3.4.2 鬼魂移动模块设计

本模块主要用来控制鬼的移动，这个部分包含三个主要函数，第一个是开始函数start函数，第二个是更新函数Update函数，第三个是碰撞检测处理函数，具体实现如下。

第一，start函数的实现过程，首先，该函数会在设置一条固定道路使其能够从中心的围墙中走出，完成游戏的初始化运行行动；

第二，Update函数的实现过程，本函数主要是进行状态更新，该函数首先会判断鬼魂是否抵达了当前道路的终点，如果没有到终点，则继续移动至道路的下一个点；如果是终点，则重新随机加载一条道路，让鬼魂朝着可以移动的方向移动过去。最后函数会把移动的方向参数传递给动画状态机。

第三，碰撞检测处理函数的实现过程，当鬼魂发生碰撞时，因为游戏规则规定了鬼魂之间的碰撞不会有任何影响，所以只需要判断鬼魂是否与吃豆人发生了碰撞即可。当其与吃豆人碰撞时，先判断是否与超级吃豆人发生了碰撞，如果是，则将鬼魂传送至起点位置，把鬼魂移动路径长度清零，最后玩家的得分还要加上500。如果是与普通吃豆人碰撞，那么游戏即可结束，因此移除碰撞体，调用Instantiate函数结束游戏，同时用Invoke函数在3秒之后运行restart函数重置游戏。

3.4.3 吃豆人移动模块设计

本模块主要用来控制吃豆人的移动，由于吃豆人的移动不是随机的，而是靠键盘输入指令来完成的，因此较为复杂，具体实现方式如下。

首先，通过插值获得下一次要移动的目标给变量dest，随后当吃豆人到达目的地时才可以接受键盘输入，然后通过GetKey函数当键盘输入了按键而且当前方向是有效的时候（即按下的是‘w’‘a’‘s’‘d’键中的一个），根据按下的键值不同，把向量不同的方向值赋给方向变量dest。

同时，我们还需要编写一个判断方向是否有效的函数，在这个函数中，首先记录当前位置，然后调用hit函数过吃豆人发射一条射线，判断射线是否打到了吃豆人自己身上，如果达到了则证明位置可达。

3.4.4 豆子模块设计

本模块功能单一，所以功能模块设计起来也比较简单，具体设计过程如下。

首先，我们需要声明一个变量代表当前玩家是否是超级吃豆人，如果不是，那么当玩家与豆子碰撞时只需要调用加分函数，然后摧毁这个物品即可；否则的话还需要额外调用一个超级加分函数，这个函数可以给玩家增加额外的分数。

3.5 数据管理说明

Unity管理这些资源时可以将这些资源放入一个Resources目录中。Resources

目录的出现是为了解决游戏资源动态加载的问题。比如说，有些游戏在刚进入界面时，一些游戏资源是不需要加载的，而需要等到玩家进入场景的时候再加载出来，但对于本次设计的吃豆人游戏，玩家在进入游戏界面时地图资源就会加载好。在使用Resources目录时，游戏开发者将一些需要动态加载的资源放进Resources目录里面，然后通过Resource.Load()函数进行资源动态加载。

在实际的资源管理中，Unity会将资源分为外部资源和内部资源。每种资源又可以分为文件尾为.dll的脚本类资源以及美术类资源。在管理的时候，Unity又会使用两个步骤，分别为序列化和meta文件的生成。

1. unity的序列化

工程中的资源，要存储到本地磁盘，那么就会通过引擎进行一步序列化的操作，序列化的实质，就是将资源对象按照一定的顺序转换成二进制文件，名叫meta文件。这个二进制文件包含了一个工程中唯一的id：guid，所有的文件都可以根据这个id来进行查询。同时，如果多个工程合并的时候出现guid冲突，还可以自己重新生成一份guid。同时其还包含了文件的导入设置:对于一般的文件，导入设置都比较简单脚本类叫MonoImporter, 资源类叫NativeFormatImporter；贴图属于需要重点关注的类型，其导入类型叫TextureImporter，里面详细的列出对该贴图的各种压缩格式，mipmaps, 类型，uv，贴图大小等等详细的设置信息。具体的meta文件如图3.4所示。

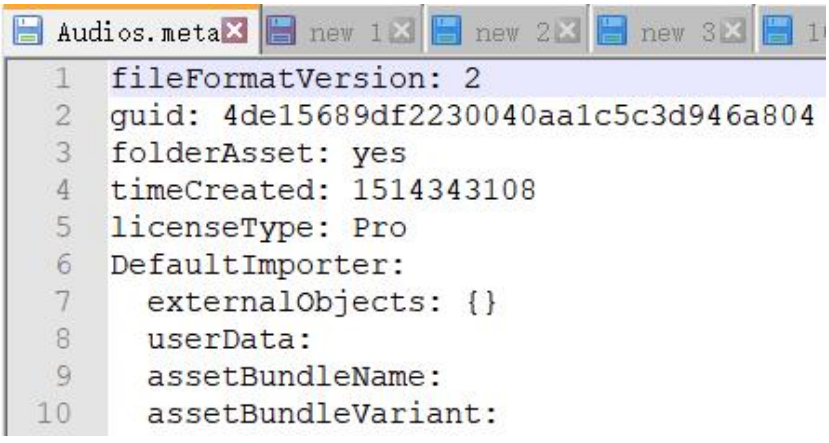

图 3.4 meta 文件实例图

2. 基于meta文件和序列化的资源管理

除了meta文件的guid，unity还会为每个资源生成一份文件id，也就是fileID,不过现在fileID已经不再保留在meta文件中了，保留到文件的序列化文件中了，

对于该资源，还会有一份localID, 这个localID, 对应的就是在一个资源中包含多个子资源的时候，定位每个子资源所用。

那么序列化是如何与guid/fileID关联的？在unity工程内部，如果给资源添加其他资源的引用，例如加一个脚本，拖拽一个外部引用，那么就会触发一次序列化操作，序列化操作的时候，就会将引用的资源的fileID和guid都序列化下来，这样在反序列化的时候，就会基于fileID和guid来反向找到依赖的资源，从而加载进来，具体如图3.5所示。

```
%YAML 1.1
%TAG !u! tag:unity3d.com,2011:
--- !u!1001 &100100000
Prefab:
  m_ObjectHideFlags: 1
  serializedVersion: 2
  m_Modification:
    m_TransformParent: {fileID: 0}
    m_Modifications: []
    m_RemovedComponents: []
  m_ParentPrefab: {fileID: 0}
  m_RootGameObject: {fileID: 1518540148733576}
  m_IsPrefabParent: 1
--- !u!1 &1518540148733576
GameObject:
  m_ObjectHideFlags: 0
  m_PrefabParentObject: {fileID: 0}
  m_PrefabInternal: {fileID: 100100000}
  serializedVersion: 5
  m_Component:
  - component: {fileID: 4110737285168328}
  - component: {fileID: 212605629630555360}
  m_Layer: 0
  m_Name: GameOver
  m_TagString: Untagged
  m_Icon: {fileID: 0}
  m_NavMeshLayer: 0
  m_StaticEditorFlags: 0
  m_IsActive: 1
--- !u!4 &4110737285168328
Transform:
  m_ObjectHideFlags: 1
  m_PrefabParentObject: {fileID: 0}
```

图3.5 guid和序列化关联图

这个过程，在Unity中，就是一个装载的过程。如果一个资源依赖的其他资源越多，那么这个装载过程就会越耗时，所以在打开一个很大的UI的时候，有一

部分的时间是消耗在装载UI上各个组件上的。

由于以上的各个过程都是在Unity内部进行的，所以在我们实际使用Unity时看到的都是这些资源的抽象形态，如图3.6所示，这些是游戏使用到的物品，图片，脚本资源等。

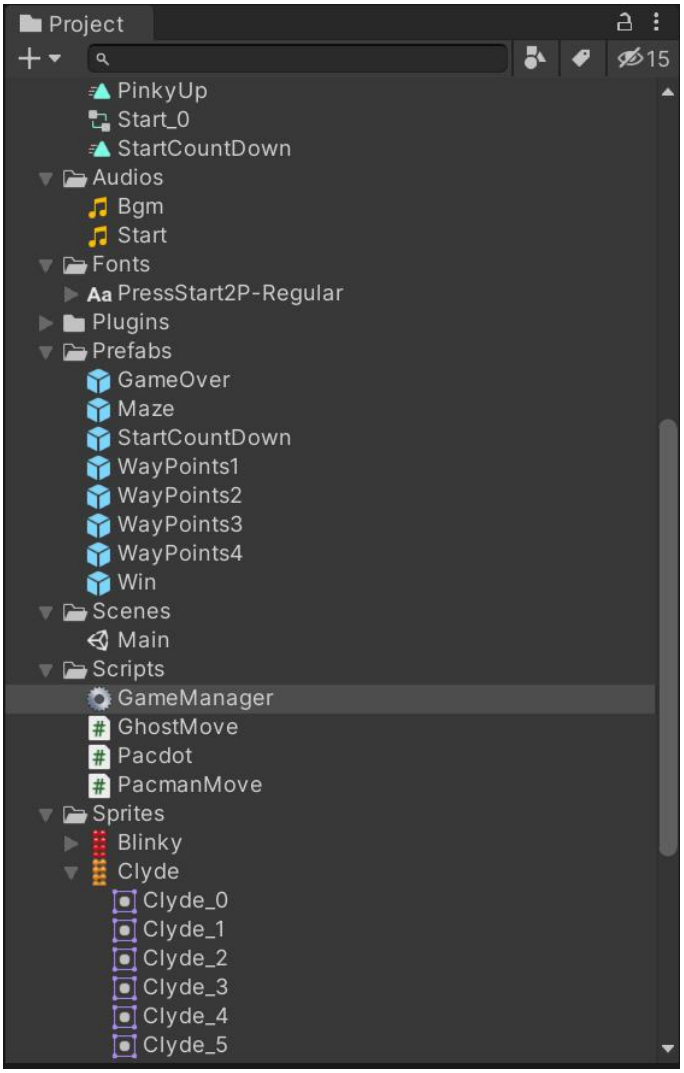

图3.6 吃豆人项目资源显示图

3. Unity对于资源的使用方式

Unity 的资源使用主要是通过引用来完成的。Unity 中的数据大部分都是引用类型，在代码中就可以对资源进行引用。每个资源也有一个引用计数，如果引用为 0，那么系统就会释放这个实体，具体加载过程如下：

（1）脚本资源的加载

在 unity 中的脚本资源，大体可以分为 C++编译的引擎 dll 文件，c#编译的 dll 文件，lua 脚本文件(基于 lua 热更的方式下)。

（2）dll 文件的加载

在工程的 Library/ScriptAssemblies 文件夹下，如图 3.7 所示，会有当前工程非引擎相关的 dll 文件列表：

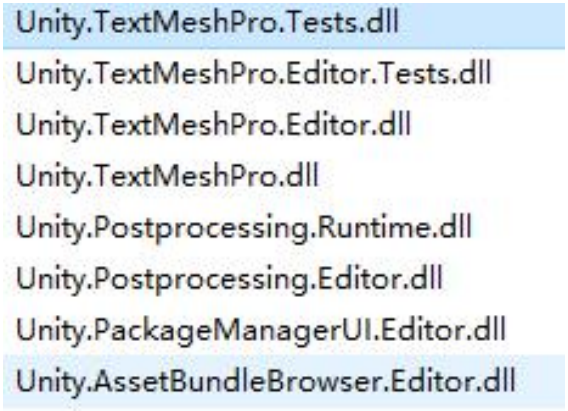

图3.7 dll文件图

在游戏启动的时候，会执行 Assembly.Load 的操作，将这些 dll 文件加载进进程中(editor 类相关的 dll 不会被加载)。

（3）lua 文件的加载

lua 由于脚本文件的属性，可以被当做类文本文件进行热更新，同时在游戏启动的时候才开启一个 Lua 虚拟机，在 Lua 虚拟机中才执行 lua 文件的 require 相关操作。这类文件的加载，其实质就是将这部分代码读入到 lua 虚拟机的全局缓存中。

（4）非脚本资源的加载和卸载

非脚本资源，才是整个游戏进程中需要处理的主要部分，会伴随整个游戏进程，直到游戏进程结束。

（5）Serialization and Instance

unity 在序列化的时候，对于每个组件，也是单独逐个的执行序列化的操作的，其序列化信息关键信息是文件本身的 fileID，以及依赖文件的 fileID 和 guid.

对应的，在 unity 的 Instance 操作中，unity 会为该 GameObject 创建一个唯一的 InstanceID, 在进程内部会缓存这样一个 InstanceID <-> gameobject 的映射关系表，同时 fileID/GUID/LocalID 会对应的映射到该文件的源文件存储位置。这个 InstanceID 具有唯一性，当 InstanceID 创建完成后，如果 object 没有被 load，则会触发 unity 执行一次资源 load，基于 fileID/Guid/local id 来执行 object 的加载。

（6）InstanceID 的创建和失效

在游戏启动的时候，会将启动场景以及 Resources 目录下的资源逐个创建对应的 InstanceID, 放入到缓存中，这部分 InstanceID 的创建耗时会随着 Resources 目录包含资源的增多而增大，所以尽量减小这部分资源的数目。

在游戏启动完成后，后续按需加载的 Object，都会对应的创建 InstanceID, 在卸载该资源的时候，会对应使这个映射关系失效，后续重新加载该 object 的时候，是会被重新创建对应的 InstanceID, 先前创建的 InstanceID 是不会被重新定位到新加载的 Object 的。

（7）AssetBundle 中文件的加载

现在在 Unity 中主要的资源管理是基于 AssetBundle 的管理，那么运行时，是如何从 bundle 中加载出想要的 Object 的？

（8）Bundle 文件的组成

在 Unity 中，bundle 会被分为两种大类，场景 Bundle 和非场景 Bundle，利用 unity 自带的 WebExtract 和 Binary2Text 两个工具，是可以解压 bundle 为文本文件的。

场景 Bundle 在使用 WebExtract 解压后，会得到两类文件：第一，BuildPlayer-sceneName: 场景序列化文件，就是 hierarchy 序列化的结果；第二，BuildPlayer-sceneName.sharedasset: 场景依赖的文件。

普通 bundle 在使用 WebExtract 解压后，同样也会得到两类文件：第一，CAB-GuidString: 该二进制文件为该 bundle 的序列化文件，以及可能包含的具体 Object 文件；第二，CAB-GuidString.resS：如果包含这个文件，则上面的文件目前是不能转换成 txt 文件的。

（9）Bundle 文件中对 script 的加载

如果 bundle 中的 object 上有对应的 script，那么在构建 Bundle 的时候，会为这个 script 构建一份特殊的资源：MonoScript，对应的存入到 bundle 中，monoscript 这种资源，并没有包含实际的运行时的代码，而是存储这个脚本的 assembly name, namespace name and class name，在装配该 Object 的时候，由于 dll 已经提前装载，所以会自动的索引到装载的 assembly/namespace/class name 脚本，然后装配到该 object 上。

4 实现与测试

4.1 实现环境与代码管理

1. 软件实现环境

本游戏项目基于游戏引擎 Unity 开发实现，编程语言为 C#语言，由于 Unity 的特性，软硬件环境对于软件的开发不构成实际影响，因此就不对软硬件进行描述。本次使用的 Unity 版本统一为 Unity 2020.1.10。

2. 代码管理

本项目的仓库地址为 https://gitee.com/li-xiang-123456/bean-eater

代码签入记录如图 4.1 所示，总共提交次数为 11 次，其中组员柳昕提交 6 次，组员黎奕辰提交 4 次，组员李响提交 1 次。组员柳昕主要负责代码的测试和部分修订，组员黎奕辰主要负责代码的编写和部分测试，组员李响主要负责系统结构的设计、界面的设计以及接口的划分。

图 4.1 码云平台代码签入截图

4.2 关键函数说明

本项目主要由四个模块脚本组合而成，其中GameManager模块实现游戏中状态的更新和设置，为其他三个模块所调用。GhostMove模块主要设置了鬼移动的一些限制和方法，PacmanMove模块主要设置了吃豆人移动的一些限制和方法，Pacdot主要设置了普通的豆子和超级豆的变化。这三个模块中调用了一些GameManager的方法。

GameManager利用Awake函数对场景进行初始化，利用start函数控制游戏的开始。在游戏开始后利用PlayStartCountDown函数显示倒计时，在倒计时结束以

后，利用Update函数进行更新当前的状态并显示对应的分数。在吃到超级豆以后利用变为超级吃豆人，同时利用FreezeEnemy冻结敌人。在函数中需要控制人物的移动时调用SetGameState函数来对人物能否移动的状态进行设置。

GhostMove在start函数调用以后，利用LoadAPath进行路径的加载，并且利用该函数为鬼进行路径的加载。利用FixedUpdate更新当前鬼可以进行移动的路径状态，如果当前的路径已经到了尽头就利用LoadAPath重新进行加载。在移动的过程中如果与吃豆人发生碰撞则根据对应的状态将鬼的位置进行重新设置。

PacmanMove在start函数调用以后，在游戏正式开始以后调用FixedUpdate根据按键来设置吃豆人的实际移动方向和位置发生移动的结果。利用Valid函数判断当前的移动是否是有效的，如果有效则移动发生，反之移动不会发生。

Pacdot在开始函数调用以后，如果被吃豆人吃掉，则将豆子移除，如果是超级豆则调用GameManager模块中的超级豆函数。

1. GameManager模块主要函数功能如下：

（1）private void Start()：该函数控制音乐的开启和停止。

（2）public static GameManager Instance()：该函数可以返回单例对象，该单例对象为其他模块复制对象提供接口，以供其进行操作和控制。

（3）private void Awake()：唤醒游戏该函数会被调用，当该函数被调用后，会对迷宫中的一些设置进行初始化，如在迷宫中添加豆子等行为。

（4）private void Update()：该函数用于判断当前是否进行到需要停止游戏的状态，如果没有则继续游戏，如果需要停止则判断当前的状态是否为胜利的状态，胜利则返回win，否则显示gameover，然后重新返回到开始场景。同时在场景中还会显示分数和豆子数量的实时变化。

（5）public void OnStartButton()：该函数可以实现按下按钮后的操作，当开始按钮被按下后，会在屏幕上显示倒计时，同时在此时播放开始状态音乐。

（6）public void OnExitButton()：退出游戏应用。

（7）IEnumerator PlayStartCountDown()：被start按键调用，显示倒计时，并且在倒计时结束以后激活场内的所有对象，10s后生成超级豆。

（8）public void OnEatPacdot(GameObject go)：该函数为吃豆子函数，本函数可以用来记录当前的分数，以及剩余豆子的情况。

（9）public void OnEatSuperPacdot()：吃到超级豆分数+200分，且变为超级吃豆人冻结所有敌人三秒，在吃掉之后的10s重新产生超级豆。

（10）IEnumerator RecoveryEnemy()：本函数的作用为冻结敌人三秒，然后进行对敌人进行解冻，同时取消吃豆人的超级吃豆人的状态。

（11）private void CreateSuperPacdot()：该函数的作用为创建超级豆子，如果当前剩余的豆子数量大于5，则该函数将随机把一个豆子变为超级豆子。

（12）private void FreezeEnemy()：该函数的作用为将鬼的状态设置为无法移动，即将其冻结，同时将鬼的颜色透明度设置为原来的70%。

（13）private void DisFreezeEnemy()：该函数的作用为将鬼的状态改为可以移动，在此函数执行后，鬼可以进行移动，将敌人的移动状态设置为可以移动。

（14）private void SetGameState(bool state)：设置场景中人物的移动状态。

2. GhostMove模块主要函数功能如下：

（1）private void Start()：该函数会将鬼的初始化位置设置为开始位置正上方的三个单位处，同时，为每一个鬼加载一条不相同的道路。

（2）private void FixedUpdate()：该函数会更新当前的状态，如果当前没有走到道路的尽头，则继续行走，否则就为其重新加载一条道路。

（3）private void LoadAPath(GameObject go)：该函数的作用为加载道路，当该函数被调用后，会同时设置道路的起始位置和道路的距离。

（4）private void OnTriggerEnter2D(Collider2D collision)：如果与鬼碰撞的是鬼则继续进行，如果是吃豆人则游戏结束，如果是超级吃豆人，则游戏继续，鬼回到初始位置。

（5）private void ReStart()：用于重新加载游戏场景。

3. PacmanMove模块主要函数功能如下：

（1）private void Start()：用于保证吃豆人在开始状态不会发生移动。

（2）private void FixedUpdate()：该函数会根据按键的方向插值获得吃豆人要移动到的位置，并且调用检验函数是否可以到达，如果可以移动到指定位置，则吃豆人将发生移动，否则吃豆人将停留在原地。

（3）private bool Valid(Vector2 dir)：检测根据按键的结果，当前吃豆人是否

可以到达对应的位置，并返回判断结果。

4. Pacdot模块主要函数功能如下：

（1）private void OnTriggerEnter2D(Collider2D collision)：检测当前遇到豆子的是否为吃豆人，如果是吃豆人则将豆子移除。

4.3 测试计划和测试用例

软件测试是保证软件质量的关键步骤，是对软件规格说明、设计和编码的最后复审，常用的软件测试方法有黑盒测试和白盒测试。白盒测试的常见方法包括：逻辑覆盖法、路径测试法。黑盒测试的内容包括Alpha/Beta测试、菜单、帮助测试、发行测试、回归测试、效能测试、负载/压力测试。黑盒测试法是根据被测程序功能来进行测试，也称为功能测试，有4种常用技术：等价分类法、边界值分析法、错误猜测法、因果图法。测试计划分为对流程的测试和对功能的测试。由于源代码开放，因此下面采用白盒测试进行模块测试。

由于本次软件主要是由四个模块构成，因此对四个主要功能模块都进行一定的测试。

GameManager模块主要功能是初始化游戏开始的界面，可以用来控制游戏中人物的状态，控制背景音乐的开关，同时可以控制游戏的开始和结束，给出游戏的相关信息等。与此同时，其中的一些函数功能作为另外三个函数模块的基本功能被调用。

GhostMove模块主要功能是控制鬼的移动路径和移动方向，对于游戏中遇到吃豆人和豆子的情况分别给出了不同的处理情况。其中鬼的移动被GameManager中的FreezeEnemy所调用。

PacmanMove模块主要功能是控制吃豆人的移动和位置的合法移动，对于游戏中遇到不同的物体和状态给出不同的状态和结果。

Pacdot模块主要功能是控制豆子和其他的物体相遇时，豆子的状态情况。

根据其中函数的主要功能进行测试的选择。由于函数中主要实现的功能是在UI基础上的可视化操作，因此对于该模块，主要测试游戏开始时音量的控制，倒计时提示和播放，游戏退出时结束的显示和音量的变化。对于其中的一些状态函数由于被其他模块中的函数所调用用来实现对应人物的，因此这些函数就与其他

模块的调用函数一起进行测试，其中，整个应用的程序控制流图如图4.2所示。

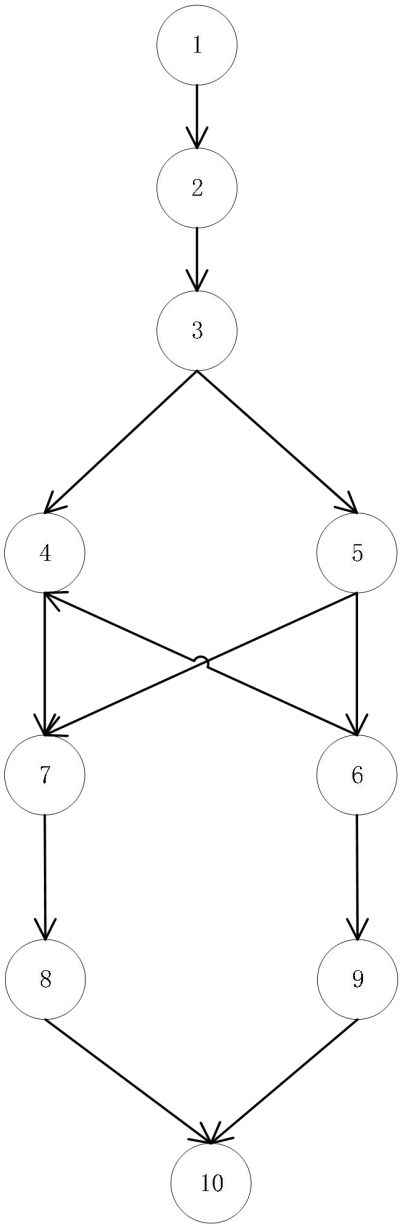

图 4.2 程序控制流图

根据如上程序的控制流图，可以给出如下的一组独立的测试路径：

1. 1-2-3-4-7-8-10
2. 1-2-3-5-6-9-10
3. 1-2-3-4-6-9-10
4. 1-2-3-5-7-8-10

下面，分别给出四条路径的测试样例和期望的测试结果，具体的软件测试路径如表4.1所示，其中四条独立路径中的所有测试样例都包含游戏时，出现的正常操作和异常操作的输入情况，因此只要完成上述的样例输入操作，就包含了所

有情况的测试。

表4.1 游戏路径测试表

路径	测试用例	期望结果
路径一	1. 在初始化界面分别点击音量的开关按钮 2. 点击start按钮，在倒计时的同时按下放下按钮 3. 在游戏开始以后，按下方向按键，在遇到障碍时按其他方向按键 4. 在游戏开始以后，吃掉所有的豆子并且不遇到鬼 5. 吃掉所有豆子获得游戏胜利	1. 背景音乐正常播放和关闭 2. 倒计时正常进行，吃豆人无法移动 3. 吃豆人可以正常移动 4. 吃豆人吃豆子获得加分 5. 取得游戏胜利
路径二	1. 在初始化界面分别点击音量的开关按钮 2. 点击start按钮，在倒计时的同时按下放下按钮 3. 在游戏开始以后，按下方向按键，在遇到障碍时仍然按该方向的按键 4. 在游戏开始后吃掉超级豆，并且在一定的时间内遇到鬼 5. 未完成任务，遇到鬼而游戏失败	1. 背景音乐正常播放和关闭 2. 倒计时正常进行，此时吃豆人无法移动 3. 遇到障碍物时吃豆人无法再继续沿同一方向进行移动 4. 超级吃豆人遇到鬼则吃掉鬼，反之游戏失败 5. 未吃完豆子，游戏失败
路径三	1. 在初始化界面分别点击音量的开关按钮 2. 点击start按钮，在倒计时的同时按下放下按钮 3. 在游戏开始以后，按下方向按键，在遇到障碍时按其他方向按键 4. 在游戏开始后吃掉超级豆，并且在一定的时间内遇到鬼 5. 未完成任务，遇到鬼而游戏失败	1. 背景音乐正常播放和关闭 2. 倒计时正常进行，此时吃豆人无法移动 3. 吃豆人可以正常移动 4. 超级吃豆人遇到鬼则吃掉鬼，反之游戏失败 5. 未吃完豆子，游戏失败
路径四	1. 在初始化界面分别点击音量的开关按钮 2. 点击start按钮，在倒计时的同时按下放下按钮 3. 在游戏开始以后，按下方向按键，在遇到障碍时仍然按该方向的按键 4. 在游戏开始以后，吃掉所有的豆子并且不遇到鬼。 5. 吃掉所有豆子获得游戏胜利	1. 背景音乐正常播放和关闭 2. 倒计时正常进行，此时吃豆人无法移动 3. 遇到障碍物时吃豆人无法再继续沿同一方向进行移动 4. 超级吃豆人遇到鬼则吃掉鬼，反之游戏失败 5. 未吃完豆子，游戏失败

4.4 结果分析

首先对路径一进行测试，之后其他路径中存在重复的状态则不再一一测试，只测试其中不同的路径即可。

在路径一中，对于Awake函数，在进入游戏的界面以后，会初始化一个分辨率为1024*768的屏幕，同时将鬼和豆子进行初始化放置在迷宫中，同时由于此时没有进入start开始环节，吃豆人和鬼都是不可以移动的，因此在进入此界面时，测试按方向键是否会使吃豆人发生移动。同时在进入到开始界面时，音乐不发生播放，只有利用音乐播放的按钮才会改变使音乐进行播放和关闭。

在测试时，进入游戏初始化界面，观察游的界面大小是否与设置的大小相同。

同时测试此时音乐的播放情况，然后，分别多次无序按下静音按钮和播放按钮后，再测试音乐的播放情况，测试此时是否都是正常播放的，具体的测试情况情况如图4.3所示。

图 4.3 界面音量测试图

此时进入游戏没有声音，游戏的界面大小符合设定的1024x768的屏幕大小。按一下音量按钮则会播放声音。此时为了测试播放的声音是否会出现重叠的现象，反复点击背景音播放按钮，发现每一次点击都会重新播放背景音乐。在点击静音按钮以后，背景音乐消失。此时再次点击播放声音按钮，背景音乐又会重新开始播放。对应的结果与预期的结果相同。

对于Start和PlayStartCountDown函数，在点击进入start函数以后会出现倒计时的场景，初始界面的开始和退出键会消失。在倒计时结束之前，所有的人物都无法移动，在计时结束后显示对应当前得分，游戏正式开始，此时人物可以移动。

在测试时，如图4.4所示，开始和退出键消失，此时按方向键吃豆人无法移动，鬼也保持不动。此时背景音乐如果有则消失，替换的是准备开始的背景音乐。

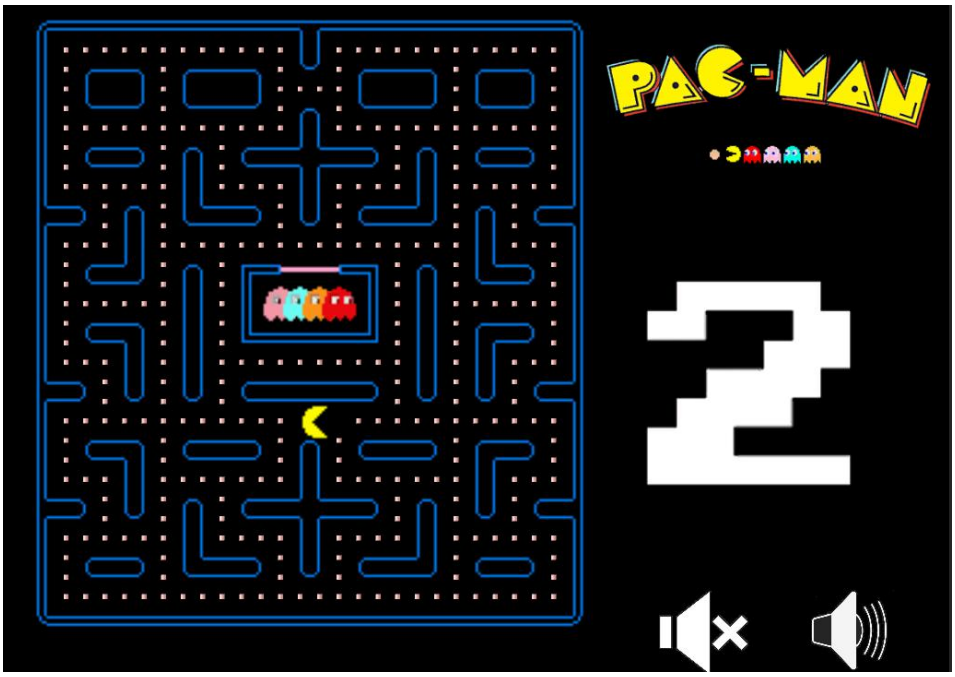

图 4.4 倒计时界面测试图

图 4.5 游戏开始界面测试图

在准备开始的背景音乐结束之后，之前的背景音乐再重新进行播放。此时的

吃豆人和鬼都可以进行正常的移动。此时如果再次点击音量的开启和关闭键依然可以正常的对音量进行开启和关闭，测试情况如图4.5所示。

在游戏开始以后吃豆人可以正常进行移动，鬼也会随机进行移动。同时吃豆人可以吃掉豆子，此时对应剩余的豆子数量和分数会实时发生变化。

此时对应音量的播放进行变化和调试，音量可以进行播放和关闭。对应的结果与预期的结果相同。

PacmanMove模块主要是处理吃豆人移动的模块。其中包括处理吃豆人上下左右的方向移动，以及对应的位置是否是吃豆人可以到达的位置。同时还有吃豆人与其他对象发生碰撞时需要进行的处理以及对应的结果。

对于Start函数，在程序按下开始按钮后就会自动执行，在按下后，由于需要倒计时，因此需保持此时吃豆人是没有办法进行移动，对应的测试如图4.6所示。

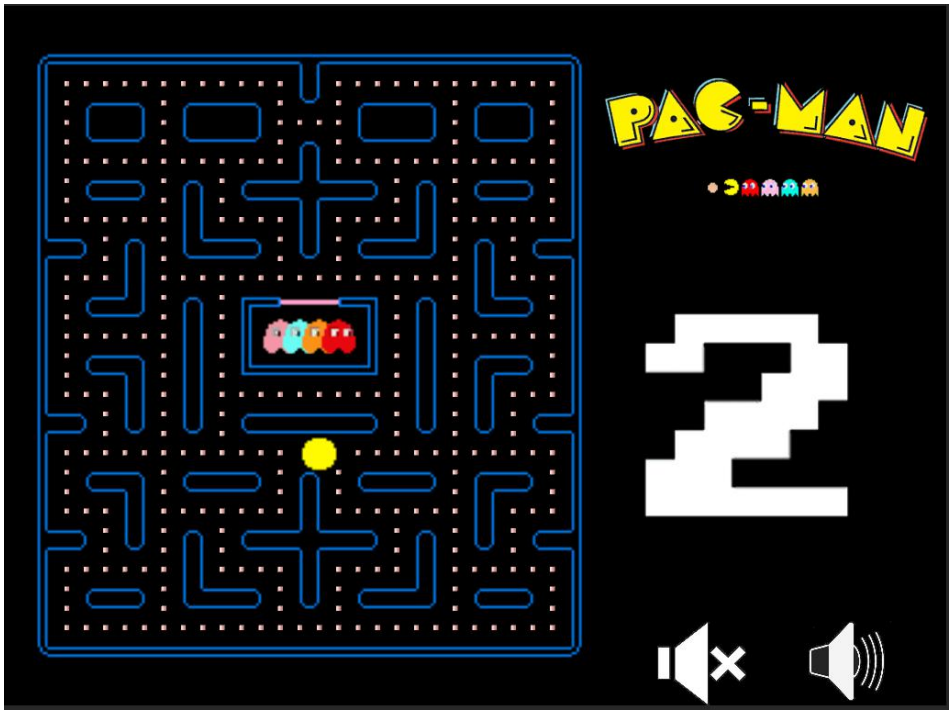

图 4.6 吃豆人移动测试图

在游戏倒计时进行时，点击方向按键，吃豆人没有发生移动，在游戏开始后，吃豆人可以进行正常的测试。

对于FixedUpdate和Valid函数，吃豆人根据按键的上下左右以及移动的位置是否可以到达，为当前的屏幕更新吃豆人的位置。主要需要测试吃豆人在吃到普通豆子，超级豆，遇到障碍物和鬼的移动结果。

当吃豆人吃到普通豆子时，吃豆人成功进行移动且当前位置的豆子被移除。

测试结果如图4.7所示。

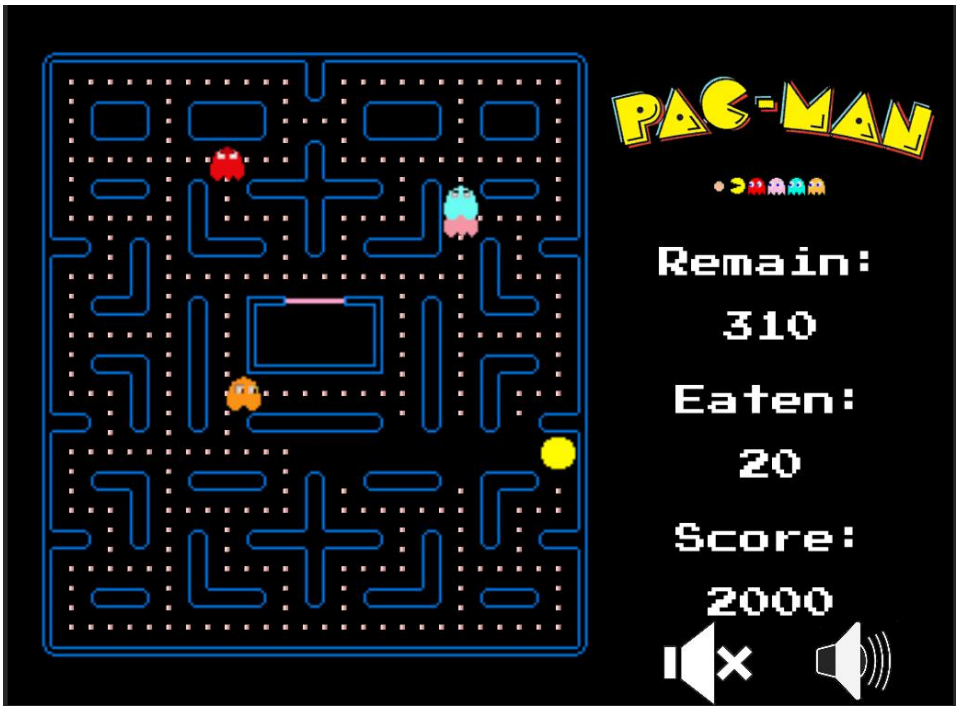

图 4.7 吃豆人吃普通豆子测试图

在吃豆人移动到有豆子的位置时，吃豆人可以发生正常的移动，在吃豆人移动以后，对应位置的豆子被移除。当吃豆人遇到障碍物时，由于前面的刚体没有办法进入，因此吃豆人就对于当前的按键位置移动不起反应，吃豆人就保持在原来的位置不发生变化，测试结果如图4.8所示。

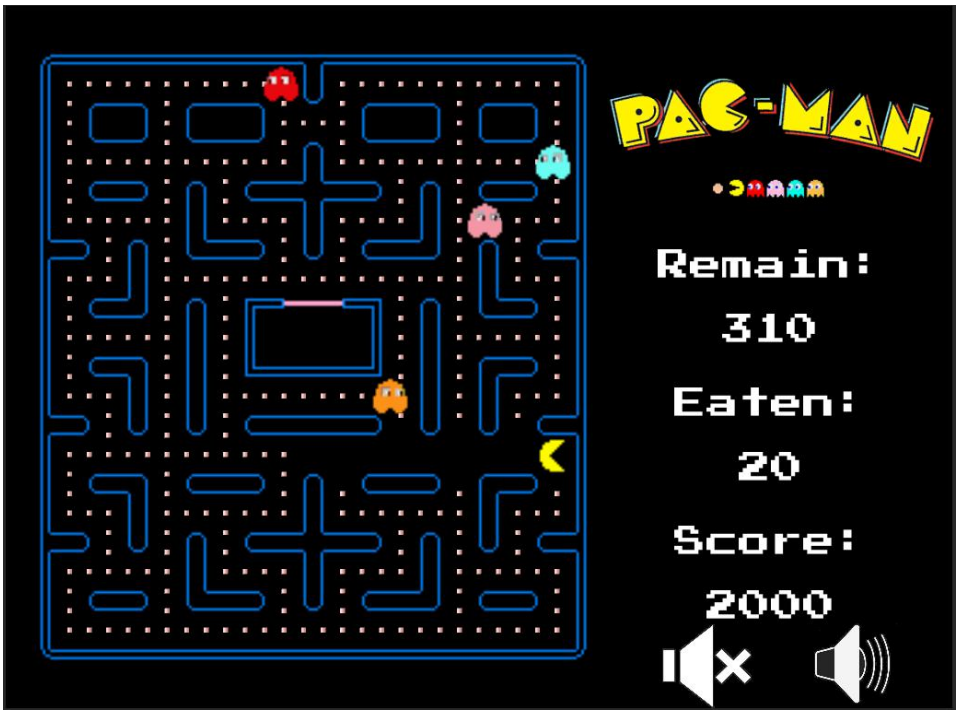

图 4.8 吃豆人与障碍物相遇测试图

在将场上所有的豆子都吃完以后，获得游戏的胜利，正常退出到初始化界面，测试结果如图4.9所示。

图 4.9 吃豆人游戏胜利图

在吃掉所有的豆子以后，游戏确实获得了胜利，因此该结果与预期相符合，对应路径一的所有测试结果与预期结果都相同。在之后的路径中，对于重复的部分就不再进行测试，只对于不同的路径部分进行一定的测试。

图 4.10 吃豆人与鬼相遇测试图

在路径二中，对于FreezeEnemy和DisFreezeEnemy函数，在超级豆子被吃掉的时后，系统就会调用冻结敌人函数，敌人被冻结三秒并且透明度降低，在时间到后，自动调用解冻功能。

在冻结期间吃豆人可以移动，但是鬼没有办法移动。吃豆人还可将给鬼吃掉，但是在吃豆人没有成为超级吃豆人前，吃豆人遇到鬼则会导致游戏结束，具体的测试结果如图4.10所示。

普通吃豆人在遇到鬼以后游戏结束，而在吃到超级豆以后，将鬼冻结，再吃到鬼则将鬼送回起始地点，测试结果如图4.11所示。

此时吃豆人遇到鬼就将鬼吃掉，送回至起始地点，同时根据分数的变化可以知道，吃掉鬼加分为500分，与加分设置的结果相同。此时按方向按键，吃豆人可以移动，但是鬼没有办法移动。在三秒钟以后，鬼恢复移动，超级吃豆人状态解除，与预期的结果相同。

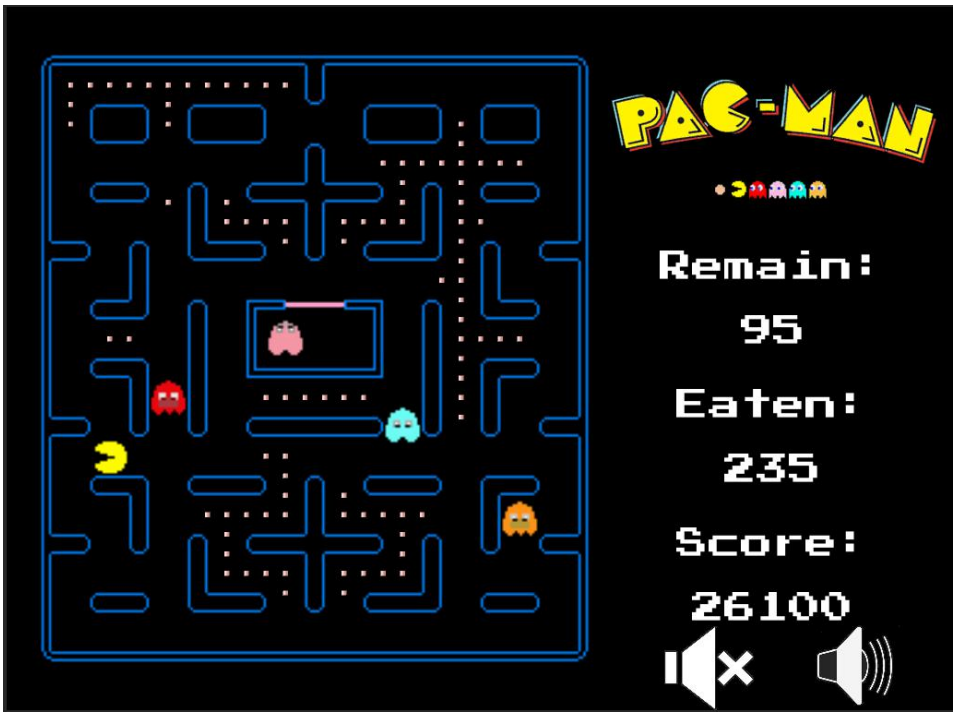

图 4.11 超级吃豆人与鬼相遇测试图

无论在游戏结束和初始化的界面，都存在exit键，对于OnExitButton函数，在初始界面点击exit按钮，程序退出游戏，而点击其他位置，不会导致游戏退出，测试结果如图4.12所示。对于正式发布版本的游戏，在点击exit按钮以后，退出游戏，与预期的结果相同。

图 4.12 退出游戏测试图

对路径三，在GhostMove模块中，在可以进行移动后，鬼根据当前加载的路径情况，和当前的行进情况随机选择方向移动。对于OnTriggerEnter2D(Collider2D collision)函数，当鬼与吃豆人发生相遇，鬼会杀死吃豆人，游戏结束，对应的测试结果如图4.13所示，在吃豆人和鬼相遇时，游戏结束，结果与预期结果相同。

图 4.13 吃豆人与鬼相遇测试图

而FixedUpdate函数用来更新当前鬼移动的状态，如果鬼走到了随机加载路的尽头，就随机重新选择一个方向进行路线的加载。其中调用LoadAPath函数进

行路线的随机加载，对应的测试结果如图4.14所示。

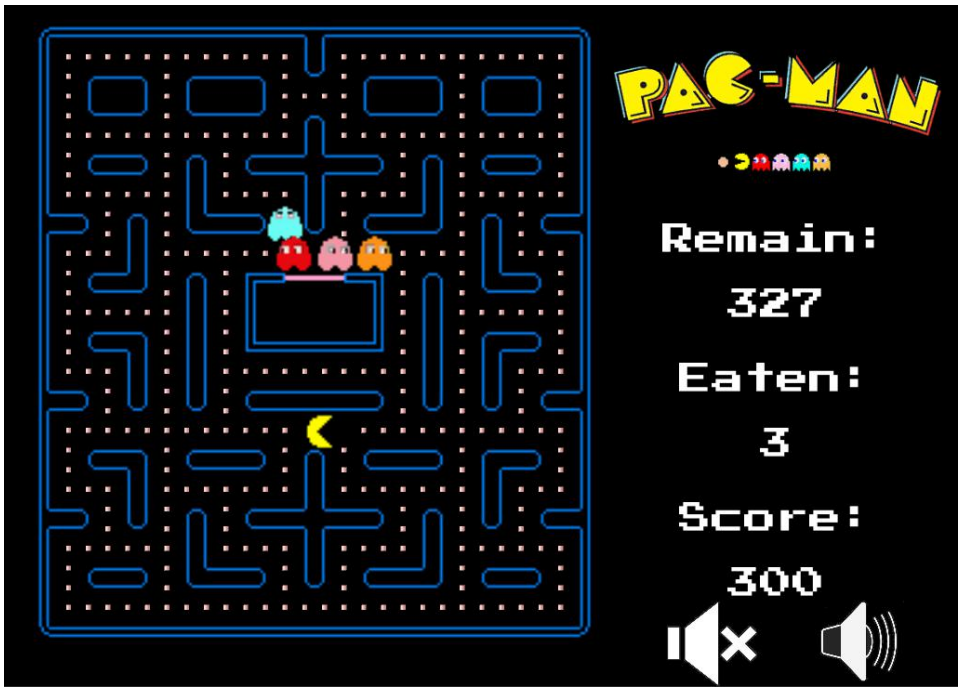

图 4.14 鬼随机加载路径测试图

对于路径四，对OnEatSuperPacdot函数和CreateSuperPacdot函数分别用来配合检测当前场上是否存在超级豆和生成一个超级豆道具。在超级豆消失的10s之后会在一个位置随机生成一个超级豆且此时豆子的数量必须大于5个，获取超级豆的加分有两百分且会对敌人产生冻结效果，测试结果如图4.15所示。

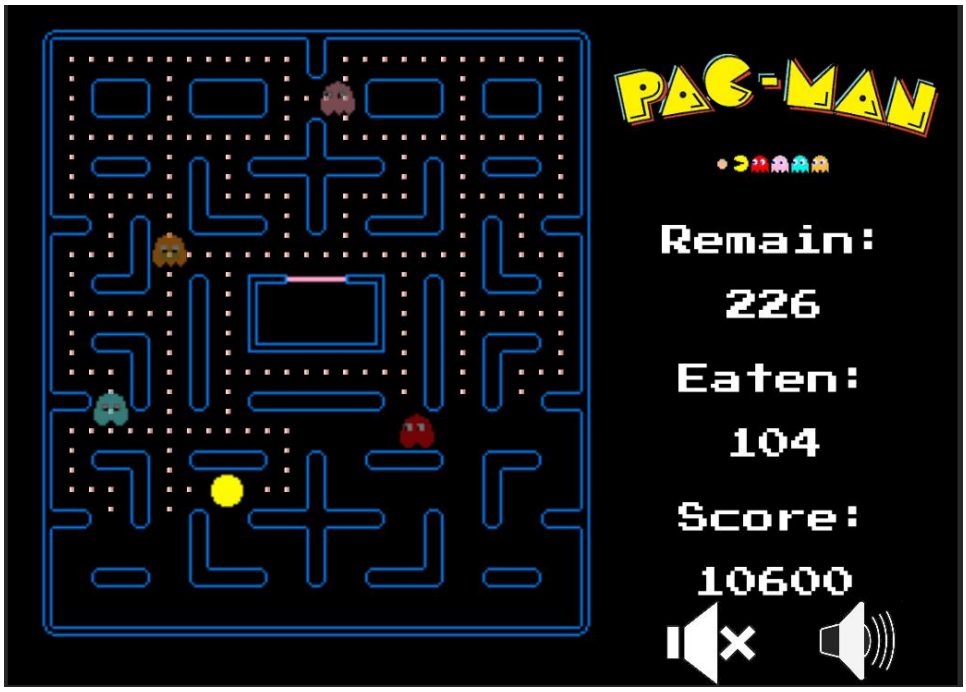

图 4.15 超级豆功能测试图

此时敌人被冻结，可以对比得分，所得的分数就是超级豆的分数为200分，因此可知超级豆分数设置正确。在吃掉超级豆以后的10s，会在场内随机生成一个超级豆。此时场上超级豆的数量大于5，生成超级豆函数没有发生错误，与预期结果相同。

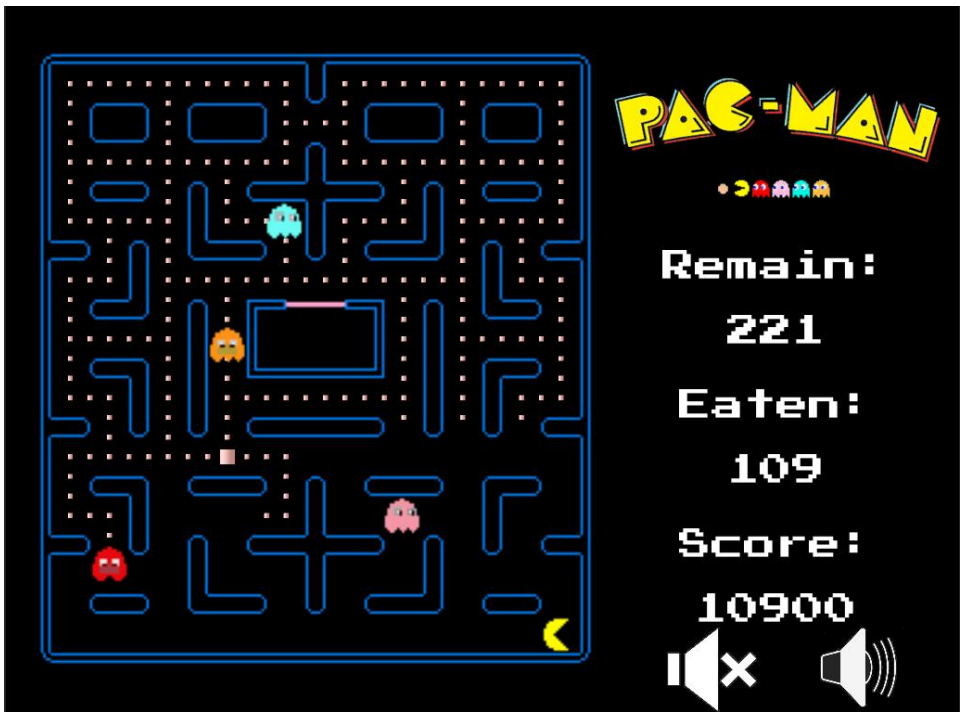

图 4.16 超级豆生成测试图

如图4.16所示，当场上的普通豆子的数量只剩下两个（小于等于5）时，在吃掉超级豆之后等待10s，并没有生成新的超级豆，因此超级豆生成函数的功能没有出现错误，与对应预期的结果相同。

在所有的路径都经过测试以后，覆盖了所有的独立路径以及所有模块中的函数，最后所有的测试结果和预期的结果都相同。

5 总结

5.1 用户反馈

本游戏项目经由推广之后吸引了不少用户玩家，得到了不少的反馈信息，经过仔细阅读，我们选了两条玩家们的反馈信息如图 5.1 至图 5.2 所示。

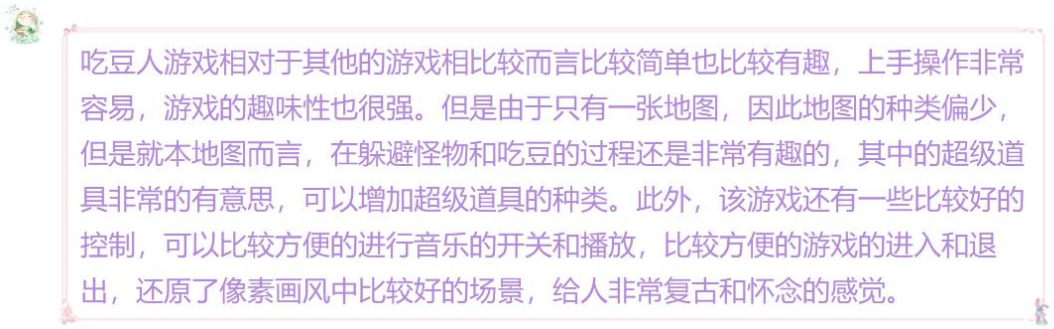

图 5.1 玩家反馈图 1

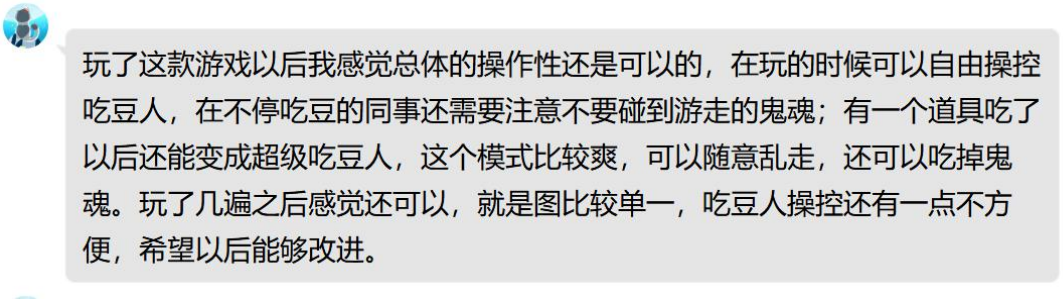

图 5.2 玩家反馈图 2

通过阅读用户的反馈信息，我们对项目有了更强的信心，我们认为该项目具有如下3条优点：

1. 操作简单，易上手。
2. 操作响应及时灵敏，趣味性较强。
3. 游戏的图形特效丰富，体验佳。

同时，我们认为该项目也存在如下可改进的地方：

1. 操作模式较为单一
2. 道具种类较少，游戏不够丰富
3. 地图固定，不能随机生成

5.2 全文总结

由于本小组的三位成员都不曾参与过大型项目的编写工作，本次软工项目是我们第一个多人合作的游戏开发项目，因此可能存在一些问题。本小组的三位成员对于此类游戏项目没有太多经验，因此，在本次项目中我们采用了民主制的组织方式设计程序和编写文档。

作为小组的一名成员，我主要负责项目前中期的项目需求分析、项目总体结构设计以及对象模型建立等工作，同时负责文档的归总和调整工作，我在本次项目的具体工作如下：

（1）完成了对于项目背景和可行性的调查和分析，使用 NABCD 模型编写完成项目背景分析，并制定了项目目标和基本计划；

（2）完成了项目的需求分析，分别使用 E-R 实体关系图、数据流图以及状态转换图，建立了项目的数据模型、功能模型以及行为模型，初步建立游戏模型；

（3）完成原型系统的架构，初步完成了原始的界面设计和实现；

（4）设计了游戏系统的总体架构，初步构建了游戏的总体架构；

（5）完成了游戏的对象模型的建立和设计，编写了对象模型设计一节；

（6）完成了项目文档的汇总和完善。

6 体会

本项目是我人生中第一次参与的实际项目，虽然遇到了许多困难，却也有许多收获。由于是我的第一个实际项目，因此遇到的困难也尤其多，虽然有些问题事后看来颇有些可笑和幼稚，但对于我来说都是不可多得的体验。

在本次游戏项目中，我所遇到的第一个问题就是，我们小组的三个成员都缺乏编写此类需要通力合作的项目的经验。所幸，在同寝室的同学中，就有一位具有大型游戏项目编写经验的同学，在他的建议和帮助下，我们小组通过商讨，顺利地选择了合适的组织方式和项目目标。我所遇到的第二个问题有关于代码的管理，由于我过去都是个人独立完成代码的编写，因此不曾使用过代码托管平台，对于代码托管比较陌生。在同学和老师的推荐下，我们最后使用了国内的一个代码托管平台码云 Gitee 进行代码的托管和多人协作。

由于我本人主要负责项目的前期分析和设计工作，因此后期的代码编写和测试方面参与不多，在项目的分析设计以文档的编写过程中，我也遇到了很多问题。

首先，是项目总体的需求分析和总体结构设计。虽然本次游戏项目所设计的是一个小型游戏，但是想要直接完成需求的分析和系统结构的设计难度依然很大。所幸，本次游戏项目的主要目标为复现经典街机游戏，系统的初步架构可以借鉴和参考原有的游戏系统，在浏览和参考了部分老游戏的系统架构，我对项目进行了初步的分析和设计，在通过进一步的分析和修改之后，我得到了最后的设计结果和系统架构等。

第二，与 Unity 的使用有关。本次项目使用 Unity 游戏引擎进行开发，虽然总体上，Unity 的使用比较简单，C 且#语言类似于 C++语言比较好上手。但是，毕竟我是第一次使用 Unity 编写程序，很多地方的使用还是比较生疏。在阅读了一些教程和小型项目编写指南后，我们很快就顺利度过了这一段比较艰难的适应过程，顺利进入了后续的代码编写过程。

第三，我负责了前期的项目需求分析和设计，也负责这一块内容的文档撰写工作，在诸如状态转换图、数据流图以及对象类图等图的绘制方面遇到了不小的麻烦，很多方面很难使用图来表现，尤其是的类图绘制。以类图的绘制为例，由于 Unity 本身的特性，大量的属性和方法使用组件进行封装和实现，很多对象无法使用类图进行清晰的描绘，只能对自主编写的脚本进行类图分析和建模。因为 Unity 本身的组件难以使用的特性，使得使用类图分析本游戏系统类的关系不够

清晰和实用。

最后，是代码编写过程中产生的 Bug。众所周知，Bug 是程序员逃不脱的一个噩梦，在本次代码编写过程中，我也遇到了各种各样的 Bug。有些 Bug 是逻辑问题，有些是对 Unity 的组织方式和运行逻辑不够了解导致，有些则是单纯对 C#语言不够熟练导致。这些形形色色、各种各样、千奇百怪的问题在借助网络资源和询问咨询同学后，最后都得到了较为圆满地解决。

在本次项目开发过程中，我遇到了很多问题，也解决了这些问题，在这个过程中，我收获了许多知识和体会，也学到了很多新的方法和技巧。

首先，在团队协作完成项目过程中，我学会了合作开发项目的一些方法，体会到了团队开发的过程和成员任务分配和协调的重要性。在项目开发过程中，虽然遇到了模块划分不清、任务分配存在问题等情况，但是经过一段时间的磨合后，我们顺利地完成了项目的开发工作。

其次，我学会了撰写报告文档的许多方法、技巧和工具，如 E-R 图、系统结构图、类图等等。这些工具可以更好地为之后的报告撰写提供帮助。

最后，在这样一个较大型软件的开发过程中，如何将软件拆分成合理的模块和子体，使得多人合作更加顺畅和合理。这些方面的技巧和知识都在之后的生产实习中具有重要的意义和作用，可以说本次项目给了我完全不同的体验。

总而言之，我在这次软件开发过程中着实收获良多，在此，我要感谢那些给予我们支持和帮助的同学和老师，尤其是我们寝室的杨彪同学，他在 Unity 使用方面给予了我很多帮助，同时，我也要感谢同组的成员们，感谢你们和我一起完成了这样一个项目。