单 位 代 码 100000

1

号



学 学 号

分 类 号

密 级



毕业设计(论文)

## 院 （ 系 ） 名 称 继续教育学院 专 业 名 称 计算机科学与技术学 生 姓 名 陈玥

## 指 导 教 师 李耀

## 2023 年 1 月

**北京航空航天大学**

**本科毕业设计（论文）任务书**

#### Ⅰ、毕业设计（论文）题目：

#### 基于Android 系统的手机益智游戏的设计与实现

#### Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

#### 原始材料数据：

#### 1.相关的C#技术资料

#### 2.相关的游戏框架资料

#### 3.相关数游戏底层技术资料

#### 技术要求：

#### 1.游戏逻辑采用C#语言编写

#### 2.Unity引擎的应用技术

#### 3.自定义渲染管线技术

#### 4.编写Shader的技术

#### 5.3D模型制作和修改，3D动作的制作和修改，3D游戏特效的制作

#### 6.2D贴图绘制

#### Ⅲ、毕业设计（论文）工作内容：

#### 1、查找相关文献资料，确定选题。

#### 2、编写毕业设计（论文）任务书。

#### 3、整理、阅读文献资料和需要参考的论文。

#### 4、思考论文结构，编写论文。

#### 5、编写摘要、目录、致谢等。

#### Ⅳ、主要参考资料：

#### 志宏.Android 应用开发详解[M].电子工业出版社.2010.[2]杨丰

#### 盛.Android 应用开发揭秘[M].机械工业出版社.2010.

#### 李宁.Android/OPhone 开发完全讲义[M].中国水利水电出版社.2010.

#### 马越.Android 的架构与应用[D].北京：中国地质大学(北京)硕士学

#### 位论文，2008:9-20.

#### 高焕堂.Android 应用框架原理与程式设计 36 技[M/OL]. Google 公

#### 司,2008:132-150.

#### 专业 计算机科学与技术 班级 一班 姓名 陈玥

#### 毕业设计（论文）时间： 年 月 日至 年 月 日

#### 答辩时间： 年 月 日

#### 成 绩：

#### 指导教师：

本人声明

#### 我声明，本论文及其研究工作是由本人在导师指导下独立完成的， 在完成论文时所利用的一切资料均已在参考文献中列出。

作者： 签字：

时间：年 月

### 基于 Android 系统的益智游戏的设计与实现

**摘 要**

游戏-作为广大人民群众的休闲和益智行为，活跃在我国的各个历史阶段。在古代，下棋、蹴鞠等行为给我国古代人民带来了无穷的乐趣，同时对于智力和体力的锻炼也带来了不少积极作用，人民一直就非常喜欢这类活动。随着科技的发展，游戏的表现形式也随之发生了巨变。由于电子设备的出现和普及，给广大人民群众带来了新的娱乐体验。

本设计是基于 Android 系统的益智游戏的设计与实现，游戏的题材和玩法多种多样，这里我选择去制作一款种地类型的益智游戏，同时带有生存策略部分。玩家需要在自己的领地内种植各种农作物，养殖牲畜，生产出日常生活所需的各种物品，同时打造出装备来抵御敌方单位的入侵，保护领地生存下去。玩家需要收集或者购买农作物的种子种在地里，然后需要浇水、施肥、驱虫来保证农作物的正常生长，在成熟之后可以进行采摘，制作成食物或者装备的材料来打造装备。通过养殖牲畜制作肉类或者收集皮毛制作装备。玩家可以建造围栏来保护领地或者防御塔来帮助玩家守护领地等。

系统使用Unity引擎来制作，使用游戏引擎可以大幅度提高制作效率。游戏内的

逻辑采用C#语言进行制作，在引擎内制作编辑器来制作和维护各种表格，保存游戏内物品的数据。角色逻辑分为两部分，主角使用有限状态机来制作，其余角色使用行为树来制作，同时制作了行为树的编辑器，使用节点形式来编辑角色的行为，大幅提高制作大量角色逻辑的效率。状态机和行为树的相关逻辑全为本人自行编写，没有使用任何插件。同时游戏内会有实时天气系统，天气分为晴天、雨天，在晴天时农作物可以正常生长，在雨天时农作物会被灌溉雨水，可能会导致农作物死亡。为了表现天气效果，需要自己编辑渲染管线，以及使用Shader来丰富画面效果。

关键词：Unity，游戏，有限状态机，行为树，Shader，序列化

## Design and Implementation of Farming Game Based on Android System

**Abstract**

Games - as a leisure and puzzle activity for the masses, have been active in various histo rical stages of China. In ancient times, activities such as chess and cuju brought endless fun t o the ancient Chinese people. At the same time, they also had many positive effects on intelli gence and physical fitness. People have always loved these kinds of activities. With the devel opment of technology, the expression form of games has also undergone tremendous changes. With the emergence and popularization of electronic devices, it has brought new entertainme nt experiences to the masses.

This design is based on the design and implementation of puzzle games based on the An droid system. The theme and gameplay of the game are diverse. Here, I choose to make a far ming-based puzzle game with a survival strategy part. Players need to plant various crops, rai se livestock, and produce various items needed for daily life in their own territory, while buil ding equipment to resist the invasion of enemy units and protect the survival of the territory. Players need to collect or buy seeds of crops, plant them in the soil, and then water, fertilize, and drive away insects to ensure the normal growth of crops. After maturity, they can be harv ested and made into food or equipment materials to make equipment. Produce meat through r aising livestock or collect fur to make equipment. Players can build fences to protect their ter ritory or defense towers to help defend their territory, etc.The system uses the Unity engine f or production. Using game engines can greatly improve production efficiency. The game logi c is made using C# language. An editor is used in the engine to create and maintain various ta bles and save data on items in the game. The character logic is divided into two parts.

The main character is made using a finite state machine, and the rest of the characters are ma de using a behavior tree. At the same time, a behavior tree editor is made using node form to edit the behavior of characters, greatly improving the efficiency of making a large number o f character logic. The logic related to the state machine and behavior tree are all written by myself without using any plugins. At the same time, there will be a real-time weather system in the game. The weather is divided into sunny and rainy days. Crops can grow normally on sunny days, while on rainy days, crops will be irrigated with rainwater, which may cause cr op death. In order to show the weather effect, you need to edit the rendering pipeline and use shaders to enrich the screen effect.

**Key words:** Android system, Farming, Game, Finite State Machine, Behavior Tree , Shader, Serialization

目 录

* 1. [绪论.............................................................................1](#_bookmark0)
     1. [研究背景](#_bookmark1) [1](#_bookmark1)
     2. [国内现状 1](#_bookmark2)
     3. [研究的目的和主要内容](#_bookmark3) 2
     4. [论文组织结构](#_bookmark3) 2
  2. [游戏制作理论以及工具介绍](#_bookmark4).........................................................4
     1. [游戏制作主要流程](#_bookmark5) 4
     2. [主流游戏引擎介绍](#_bookmark6) 4
  3. [C#语言特点](#_bookmark6) 4
  4. [Shader语言特点](#_bookmark6) 4
  5. [3DMax和PhotoShop的介绍 4](#_bookmark6)
  6. [本章小结](#_bookmark7) 5
  7. [系统需求分析](#_bookmark8).....................................................................6
     1. [系统总体需求描述](#_bookmark9) 6
     2. 系统模块功能需求 7
     3. 数据库 E-R 模型分析 7
     4. [开发环境与硬件需求](#_bookmark6) 8
     5. [本章小结](#_bookmark0) 9
  8. 系统概要设计....................................................................10
     1. 系统总体框架设计 10
     2. 系统工作流程设计 11
     3. 本章小结 12
  9. 系统详细设计....................................................................13
     1. 表编辑器模块设计 13
     2. 行为树模块设计 14
     3. 有限行为状态机模块设计 16
     4. 管理器模块设计 18
     5. [渲染模块设计](#_bookmark10) 20
     6. 开始游戏模块设计 22
     7. [场景管理模块设计](#_bookmark10) 24
     8. 背包模块设计 26
     9. [自定义农场模块设计](#_bookmark10) 28
     10. [种植农作物模块设计](#_bookmark10) 30
     11. [农作物生长模块设计](#_bookmark1) 33
     12. [序列化与反序列化](#_bookmark6) 34
     13. [数据库设计](#_bookmark0) 36
  10. 系统实现..............................................................40
  11. [总结与展望 64](#_TOC_250002)

[致 谢 65](#_TOC_250001)

[参考文献 66](#_TOC_250000)

1 绪论

本章节主要介绍了项目的选题来源，目前现有的主流种地游戏的发展情况以及优势缺陷，最后介绍了论文的主要工作内容和论文的结构安排。

* 1. 研究背景

随着计算机和网络的日趋普及，网络在人们的生活中占据了不可替代的地位。根据中国互联网信息中心 CNNIC(China Intenet NetWork Information Center)发布的《第28 次中国互联网络发展状况统计报告》显示，截止到 2011 年 6 月底，我国网民规模达到了 4.85 亿，其中手机用户为 3.18 亿。显然手机购物必将成为继 PC 购物后又一比较流行的购物方式。手机购物相对 PC 购物方式来说更加的简单、快捷、操作方便， 人们不必被限制在电脑前购物，也不用刻意的找时间去上网购买商品，只要拥有一款Android 智能手机，你就可以随时随地的查看商品，这很大程度上来说大大的增加了购物的效率。近几年随着中国移动互联网的不断发展，数字通信技术得到了更好的推广，数字通信技术为移动终端用户带来了更快的数据传输速率，这就更加推动了手机购物成为又一流行的购物渠道。因此，开发一款基于 Android 系统的手机网上商城就是在这种背景下应运而生的。

* 1. 国内现状

目前国内各大网站都有自己对应的手机客户端，他们开发这些软件的原因也是顺应手机购物的流行趋势，抓住手机网上购物的消费者，从而更好的提高自己的销售额度。毕竟手机购物的便利性是远远高于 PC 购物模式的，所以某些品牌商开发的手机客户端也逐渐得到消费者的认可。国内优秀的手机客户端譬如京东，蘑菇街，淘宝， 天猫等均受到了了消费者的喜爱，用户基群大，数据交互庞大。尽管国内已经有了很多优秀的 app 客户端，但是从客户端的不断更新，功能上推陈出新，以及用户提出的使用习惯，需求，货品种类更新来考虑，开发设计一个更加新颖的 app 永远是这个市场最需要的。

* 1. 研究的目的和主要内容
     1. 研究的目的

题目要求设计并实现一个基于 Android 平台的益智游戏。用户可以通过

Android 系统手机运行该游戏，在游戏内种植各种农作物，养殖牲畜，抵抗入侵

，在一块未知的大陆生存下去。

在现代社会，生活节奏越来越快，人们对于娱乐、休闲的需求越来越多， 我们每个人都需要在忙碌的日常生活中透透气，开心一下，调整一下自己的生活状态，在各种娱乐方式中，游戏是成本最低且最容易获得的休闲方式。同时对于青少年们，适度游戏也可以提高青少年们的智力水平。

* + 1. 研究的主要内容
       1. Unity引擎配置游戏资源
       2. 使用四叉树实现游戏的场景管理
       3. 实现角色的移动和物理碰撞
       4. 实现背包功能，可以在背包内添加、使用物品
       5. 实现有限状态机，完成从用户输入到角色实现对应的逻辑
       6. 实现行为树和对应的编辑器，实现使用可视化的方式来编辑逻辑
       7. 实现玩家种地，养殖的逻辑
       8. 实现玩家制造物品的逻辑
       9. 实现装备背包，可以装备各种装备提升属性
       10. 实现建造各种建筑物的逻辑，同时实现建筑物的功能
       11. 实现与敌对单位战斗的逻辑
       12. 实现对于渲染管线的编辑
       13. 使用Shader提高游戏内效果表现
  1. 论文组织结构

按照软件开发的生命周期过程，论文整体分为七个章节。

第一章 阐述了论文的研究背景，目前国内对于手机休闲游戏的一个发展状况， 并且对基于 Android 系统的手机休闲游戏研究的目的和主要内容进行了说明。

第二章， 介绍了游戏引擎的优点以及引擎的使用技术。第三章， 针对系统要实现的功能点进行调研和分析。第四章， 对系统的基本流程和框架等做了详细叙述。第五章， 主要介绍了系统各功能模块的详细设计。

第六章， 在 AndroidStudio 开发平台下，对系统各功能模块进行编码实现。

第七章， 总结论文的主要内容，所用到的技术点，以及该系统存在的一些不足。

2 游戏制作理论以及工具介绍

* 1. 游戏制作主要流程

1972年，世界上第一款电子游戏《Pong》诞生，由雅达利公司制作。到现在为止， 游戏产业已经经过了50余年的发展，已经拥有一套非常成熟的工业化制作流程。在游戏研发中，一般先由策划制作策划文案，然后交由美术去制作美术资源，再交给程序去制作游戏逻辑。分工明确、各司其职。在公司内制作游戏主要还是会使用一些游戏引擎来制作游戏，可以大大提升游戏的制作效率，降低制作门槛。

* 1. 主流游戏引擎介绍

游戏引擎主要分为商业引擎和免费开源引擎。商业引擎有专门的商业公司负责开发、维护更新，并且对游戏制作团队提供技术支持，需要收取一定的费用。免费开源引擎一般为个人或者爱好者进行制作，开源、免费，但是一般来说引擎性能参差不齐，后续更新也是一个大问题。对于国内的公司来说一般会使用Unity引擎或者Unrea l引擎来进行游戏开发，前者更适合移动端，后者更适合做端游或者主机游戏。

我在这个项目里使用的是Unity引擎，更适合小团队开发。

* 1. C#语言特点

C#是一种跨平台、面向对象、泛型编程的计算机编程语言，在Unity引擎的游戏开发中使用该语言。C#语言吸收了 C++、VB、Java等语言的特点，体现了当今最新的程序设计技术的功能和精华。去除了很多C++复杂、冗余的设计，并添加了GC垃圾处理机制，底层会帮开发人员智能的管理内存，不会造成内存泄漏等问题。语言简洁，快速应用开发功能，语言的自由性，强大的Web服务器控件，并且支持跨平台。

* 1. Shader语言特点

Shader语言也就是着色器语言，是一种专门用来为着色器编程的语言。现今主流S hader语言主要包括DirectX的HLSL（High Level Shader Language）和GLES的GLSL（O penGL Shader Language）两种Shader语言。两种语言非常类似，Unreal和Unity都使用了HLSL语言，且支持了跨平台编译，开发者们只需要编写HLSL语言的Shader即可发

在Android、IOS、PC等多个平台，大大提升了开发效率

* 1. 3DMax和Photoshop的介绍

3DMax是AutoDesk公司旗下的3D资源生产软件，能够用来制作可以在游戏引擎中使用的3D模型，通过对模型的顶点编辑、分配UV显示对应纹理、绑定骨骼实现骨骼动画等方法，制作出游戏内需要的3D模型资源，是游戏制作不可或缺的美术工具。

Photoshop是Adobe公司旗下的照片、图片处理工具，可以生成和处理2D纹理， 绘制模型的贴图，制作出栩栩如生的模型的纹理。

游戏制作中往往需要使用这两种甚至更多工具来配合制作出游戏美术资源。

* 1. 本章小结

本章主要介绍，游戏的制作流程，为什么使用游戏引擎以及游戏引擎的选择以及游戏制作所需要的语言和工具。游戏逻辑采用 C#语言编码实现，渲染效果使用HLSL Shader语言，在美术资源的制作上使用3DMax和Photoshop两款工具。

1. 系统需求分析
   1. 游戏的整体框架描述

一个基于 Android 平台的农场休闲单机游戏，该游戏由游戏资源管理器，游戏玩法逻辑，游戏的存储和加载以及渲染功能四大模块。

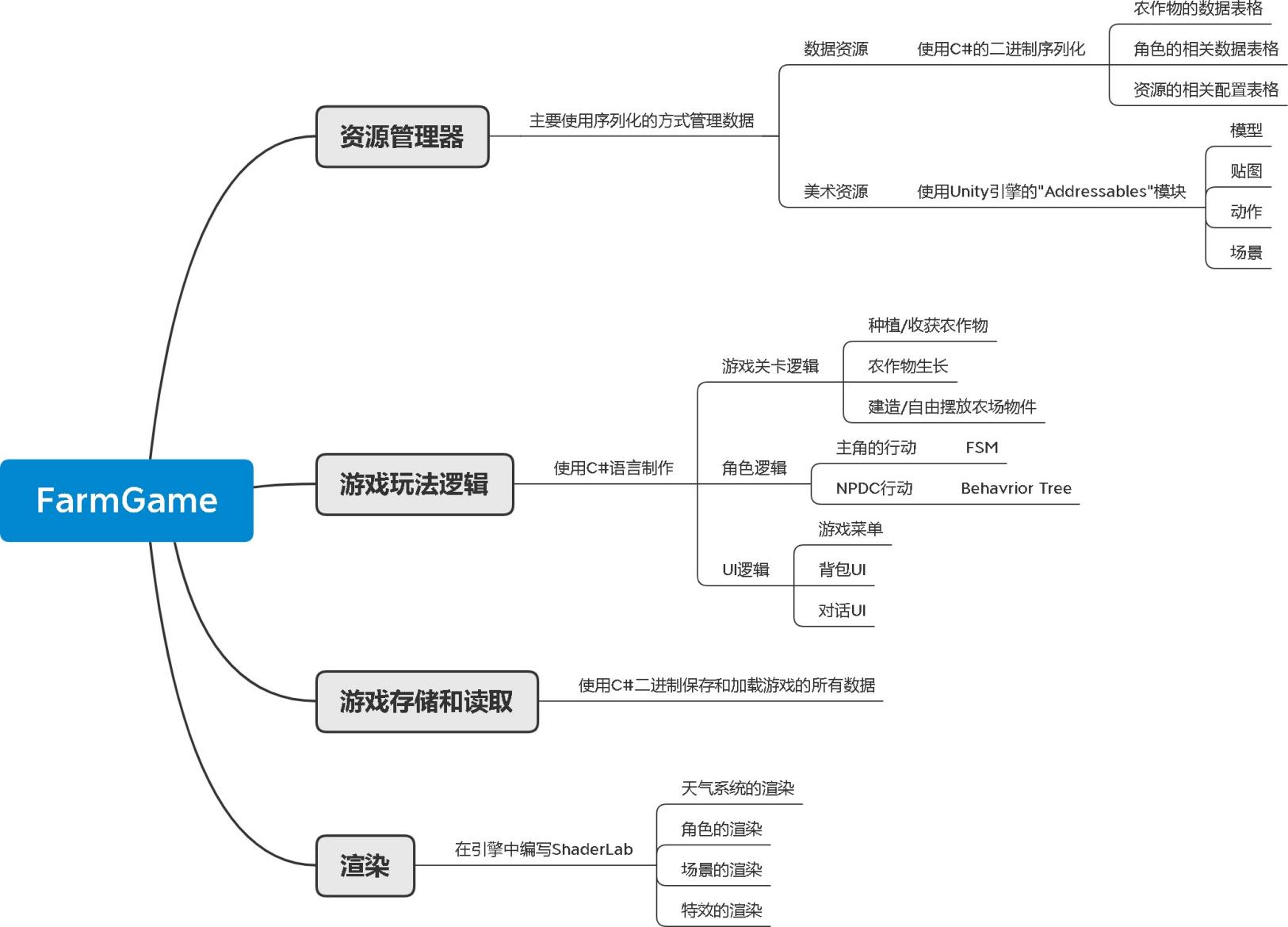


图 3-1 游戏的整体框架

* 1. 系统模块功能需求
     1. 游戏场景编辑功能，开发者使用3Dmax、PS等工具，制作出美术资源，并在引擎中编辑游戏场景，编辑和设计游玩区域。
     2. 资源的管理功能，开发者通过配置表格，并开发相应的逻辑功能，来管理加载需要的美术资源、数据资源。
     3. 自定义农场功能，将农场土地栅格化，把土地分割成N\*M个格子，玩家可以编辑每个格子的用处，包括耕种用地，放置建筑物用地，草地等等。不同的地块用于不同的功能。
     4. 种植农作物功能，玩家将种子种在耕种用地里，通过浇水、施肥驱虫等动作，保证农作物健康生长，最后收获农作物。
     5. 玩家的行为功能，玩家输入指令，通过FSM（有限状态机）来响应玩家的指令，并做出相应的行动，包括移动、耕种、浇水、施肥、收获等等行为。
     6. NPC的行为功能，使用行为树逻辑，通过节点式图形化的方式来制作各种各样的逻辑，适合制作大量的逻辑行为，适合用于制作游戏中NPC的逻辑行为。
     7. UI功能，包括多种与玩家交互的逻辑，包括系统菜单、道具背包、NPC对话、购买道具等等的UI逻辑
     8. 游戏的保存和加载功能，用来保存和加载玩家一段时间的游戏进度，玩家可以自由选择自己合适的游戏时间去游玩。
     9. 游戏的渲染功能，把游戏的资源显示在显示设备上。
  2. 游戏数据表格E-R 模型分析

基于游戏客户端需求中的各项功能，数据库需要 5张表来存储相关信息。分别是：

资源信息表，游戏道具表，自定义农场表，农作物生长配置表，对话剧情表。

资源信息表，保存游戏资源的相关信息，用来在游戏中动态加载和管理资源。游戏道具表，保存游戏中道具的相关信息。

自定义农场表，保存自定义的农场的栅格化信息。农作物生长配置表，保存农作物生长的相关信息。

对话剧情表，保存玩家和NPC的对话以及剧情的文字信息。

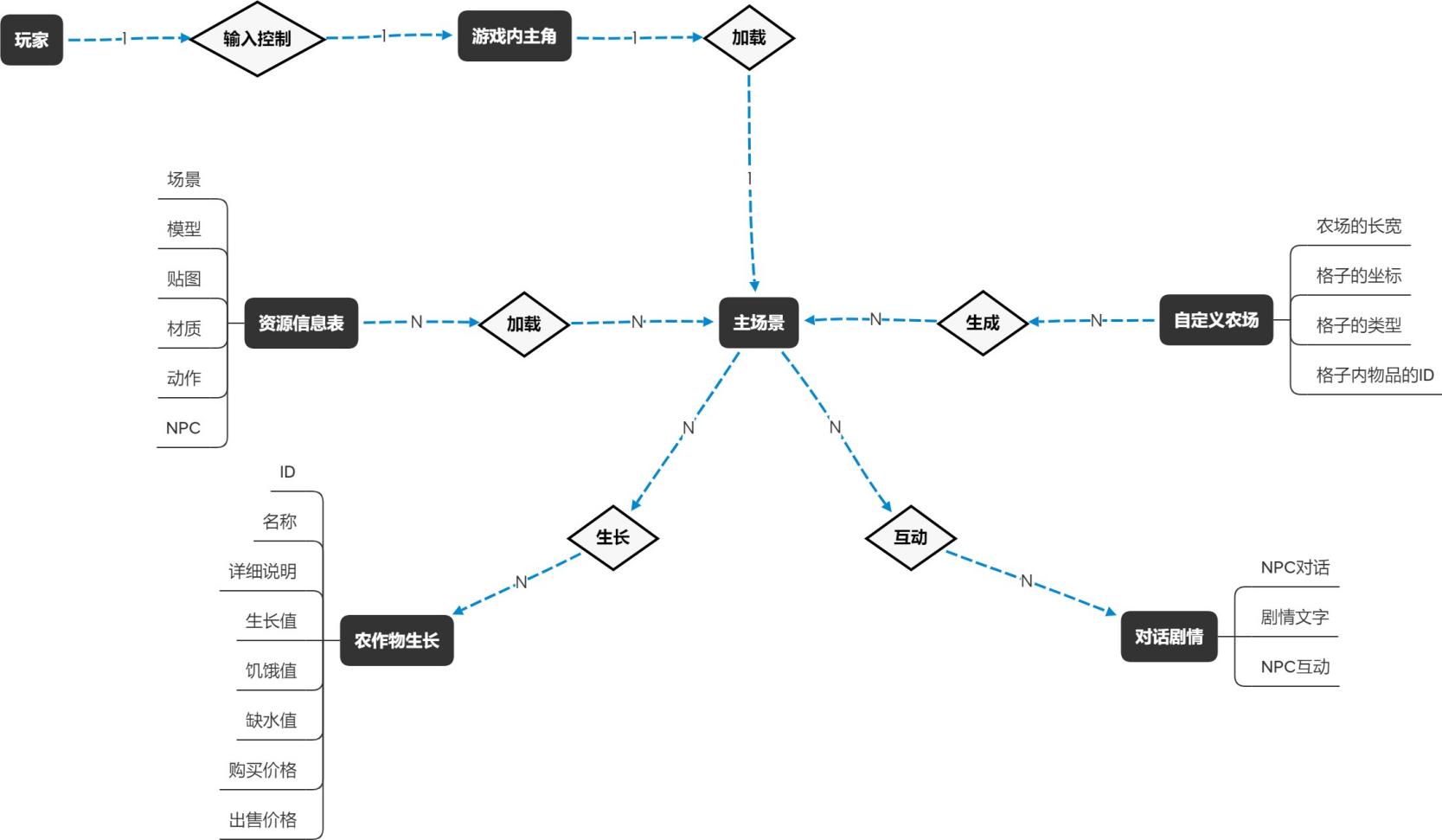


图 3-2 游戏E-R 模型

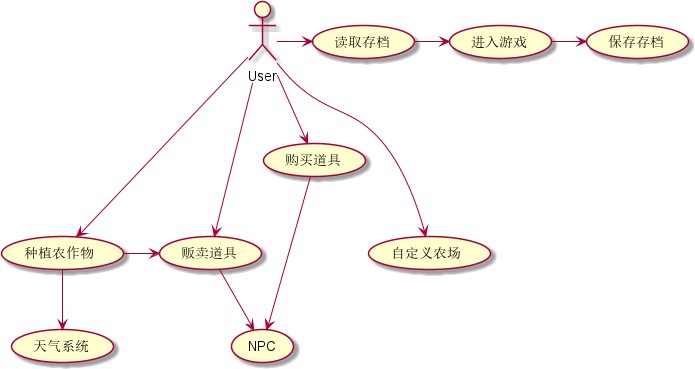


图 3-3 游戏用例图

* 1. 开发环境与硬件需求

软件类需求：

（1）操作系统：Windows 7 64bit及其以上操作系统。

（3）游戏引擎：Unity 2020 3.46f1c1

（2）美术资源：3DMax2012及以上，PhotoshopCS6及以上。

（3）开发工具：Rider 2020。

（4）开发语言：C#、HLSL。

（5）移动端发布组件：Android Build Support模块。

硬件类需求：

1. 电脑内存：8G 以上。
2. 电脑 CPU: 主频3G hz，四核以上。
3. Android 系统测试真机一部：系统 6.0 以上。
   1. 本章小结

本章节对手机网上商城系统的需求进行分析和调研，主要从以下三个方面分析： 系统功能需求、数据库需求、开发环境与硬件需求。

1. 系统概要设计
   1. 系统总体框架设计

该系统分为编辑器和运行时两部分。编辑器功能主要包括表结构的编辑工具、行为树的节点编辑工具、脚本的自定义编辑器界面等功能。运行时功能主要包括底层的多种全局管理器（资源管理器、天气管理器、角色管理器、场景管理器、渲染管理器等），以及上层的多种行为逻辑（UI逻辑、主角FSM、行为树逻辑、自定义农场编辑逻辑，农作物生长逻辑等）。



图 4-1 系统总体架构图

* 1. 系统工作流程设计

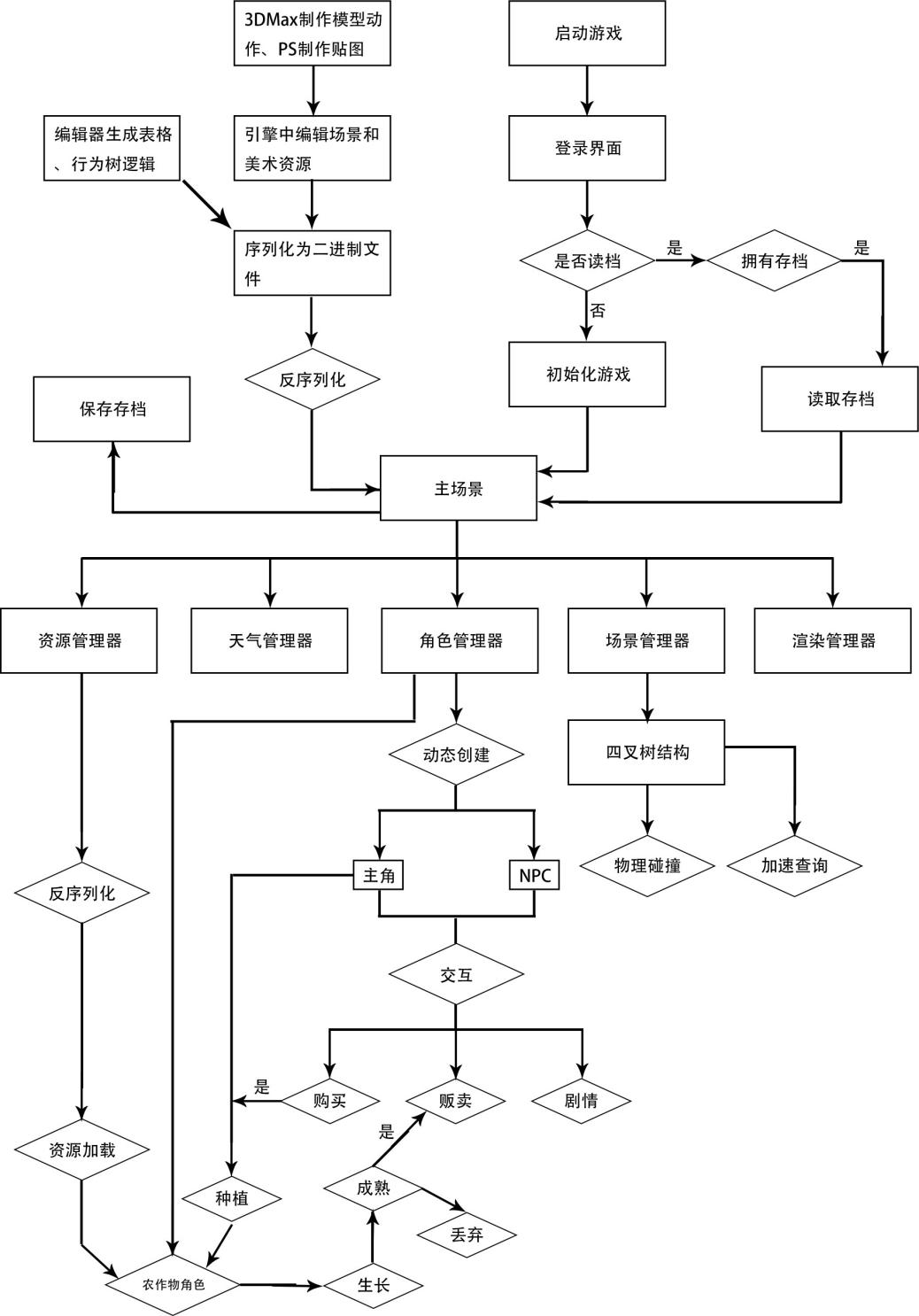


图 4-2 系统工作流程图

系统工作流程描述：

使用3DMax、PS制作美术资源，导入Unity中，使用引擎编辑游戏场景，制作表格编辑器、行为树编辑器，把生成的表格和行为树逻辑文件、场景文件、资源文件序列化为二进制文件保存在硬盘中。

启动游戏，进入登录界面，询问是否读档，选择是之后查找存档文件，如果拥有

则读取存档，没有则初始化游戏，然后加载主场景，反序列化资源文件并加载需要的文件。

初始化各个管理器，从角色管理器中创建主角和各个NPC，主角可以和NPC进行多种交互行为。当购买并拥有种子时可以进行种植活动，从资源管理器中加载资源，从角色管理器中生成农作物角色，农作物角色执行自身的逻辑进行生长，当生长完成时可以收获并贩卖，如果没有成熟则可以丢弃。

场景管理器使用四叉树结构管理场景物件，负责角色和场景地表、阻挡物进行碰撞计算，同时提供加速查询物件的功能，例如获取角色当前地块。在主场景中随时可以保存游戏进度。

* 1. 本章小结

本章主要介绍模拟农场游戏的整体框架设计图以及游戏系统的整体工作流程图。

1. 系统详细设计
   1. 表编辑器模块设计

表类继承自ITableEditor接口，该接口主要包括一些编辑布局用的字段和函数， 包括Dirty、Name属性，绘制界面用数以及各种绘制控件函数。其中：

Dirty是一个只读属性，返回当前表是否被修改，被修改的表会在退出编辑器时提醒编辑人员是否保存。

Name是一个只读属性，返回当前表的名称。

表编辑器模块时序图流程为：1.研发人员制作编辑器工具和界面，2.编辑表的数据，3.将表数据保存至数据库，4.游戏逻辑向数据库中查询表数据，5.数据库返回表数据，6.游戏逻辑中加载表数据并生成对应逻辑

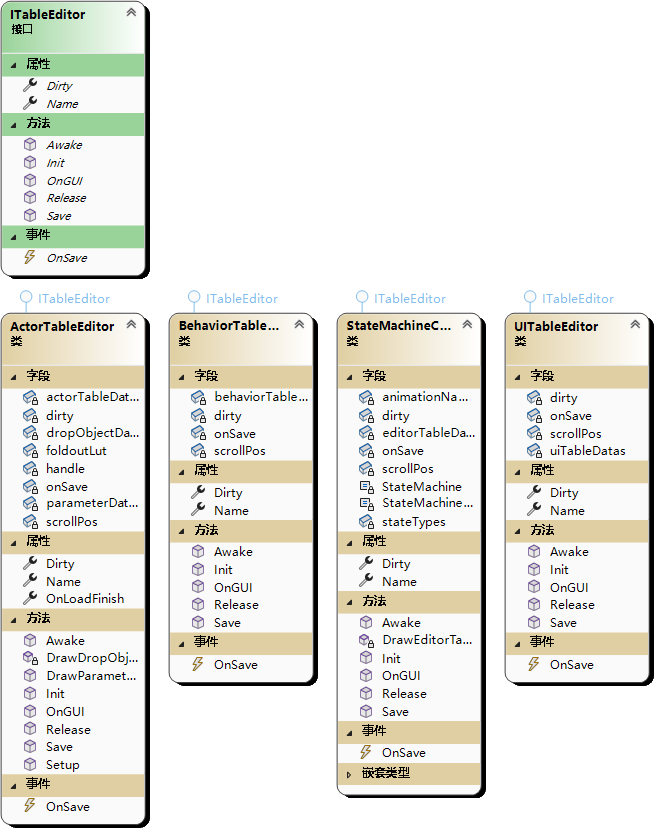


图 5-1 表编辑器类图

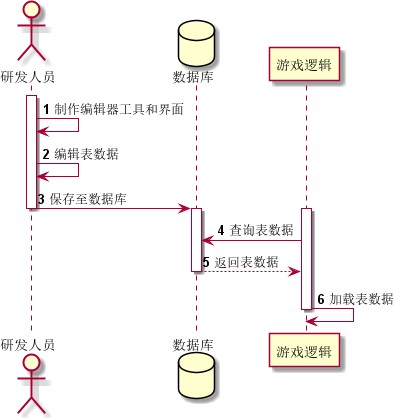


图 5-2 表编辑器和表逻辑时序图

* 1. 行为树逻辑模块设计

行为树逻辑模块主要使用在编辑器中生成的行为树资源，生成对应的逻辑节点， 使用逻辑节点生成多个具有不同调用逻辑和调用时机的行为类。资源中保存的行为树需要的数据会被用来初始化行为树逻辑，逻辑本身需要通过派生多个行为类来实现行为本身。

行为树的逻辑节点继承自BehaviorTreeNode节点，主要分为RootNode，Conditio nNode，SwitchNode，ControlNode，ActionNode五类节点。

RootNode为根节点，负责逻辑的输入工作。ControlNode为控制节点，用来控制逻辑流的走向。

SwitchNode为开关节点，控制特定情况下该节点的后续节点的开关情况。

ConditionNode条件节点，当满足某个条件，才会执行该节点的子节点ActionNode行为节点，表示角色所执行的具体逻辑

行为树模块时序图流程为1.GameLogic中每帧执行行为树逻辑，2.从RootNode中向下遍历树结构，寻找可以执行的节点，3.执行ConditionNode之后的ControlNode节点， 4.执行开关节点行为，5.判断开关节点是否为打开状态，6.执行后续的子行为节点，7. 执行子行为节点的行为逻辑，8.退出行为节点，返回游戏主逻辑

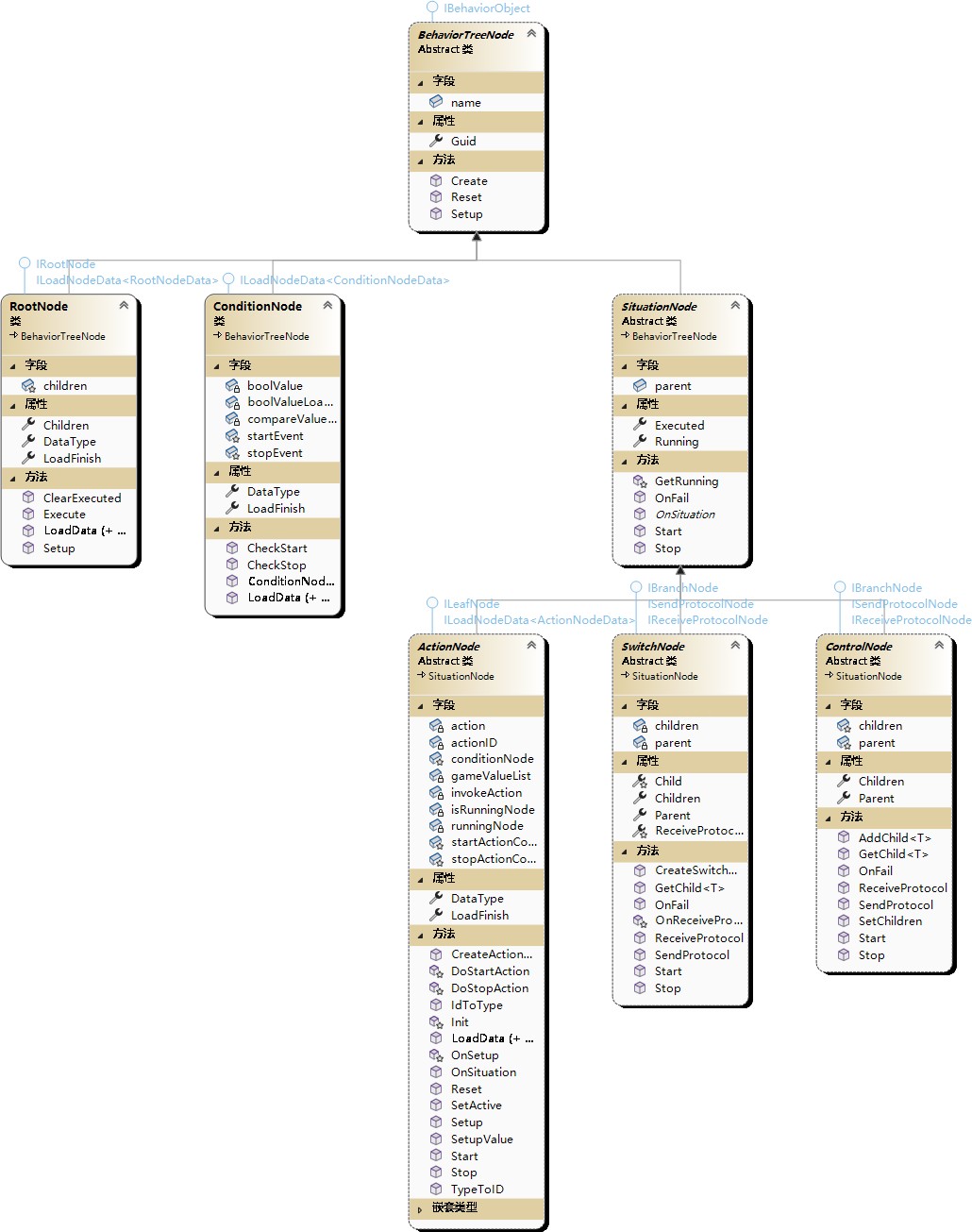


图 5-4 行为树模块类图

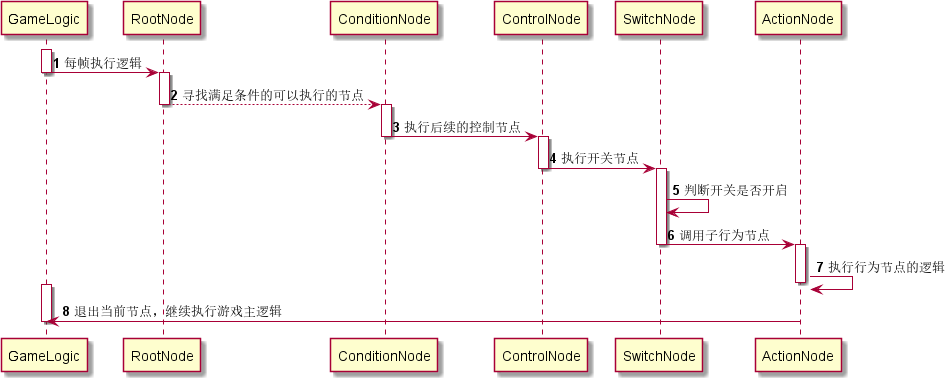


图 5-5 行为数模块时序图

* 1. 有限行为状态机模块设计

状态机功能主要实现主机逻辑，相比于NPC逻辑，主角逻辑单一且复杂，适合使用有限行为状态机来制作。继承自IState接口，通过StateSelector处理进入和退出某个State。包含banlist、requirements，onEnter，onLeave字段，PlayerStateType、Su bStateID标签。其中：

banList字段是State的黑名单，当前State无法进入处于banList中的State。requirements字段是一个delegate列表，保存了进入该State的所有条件。

onEnter是一个delegate，当进入该State时会执行。onLeave是一个delegate，当离开该State时会执行。

PlayerStateType一个只读的标签，用来返回当前State的Type。

SubStateID是一个只读的标签，用来返回当前State的子StateID。

有限行为状态机模块时序图流程描述：1.玩家对设备进行输入操作，2.状态机根据玩家操作类型选择对应的StateSelector，3.StateSelector判断当前State能否退出， 4.StateSelector判断目标State是否可以进入，5.进入目标State，6.执行目标State 中的行为，7.返回游戏主逻辑。



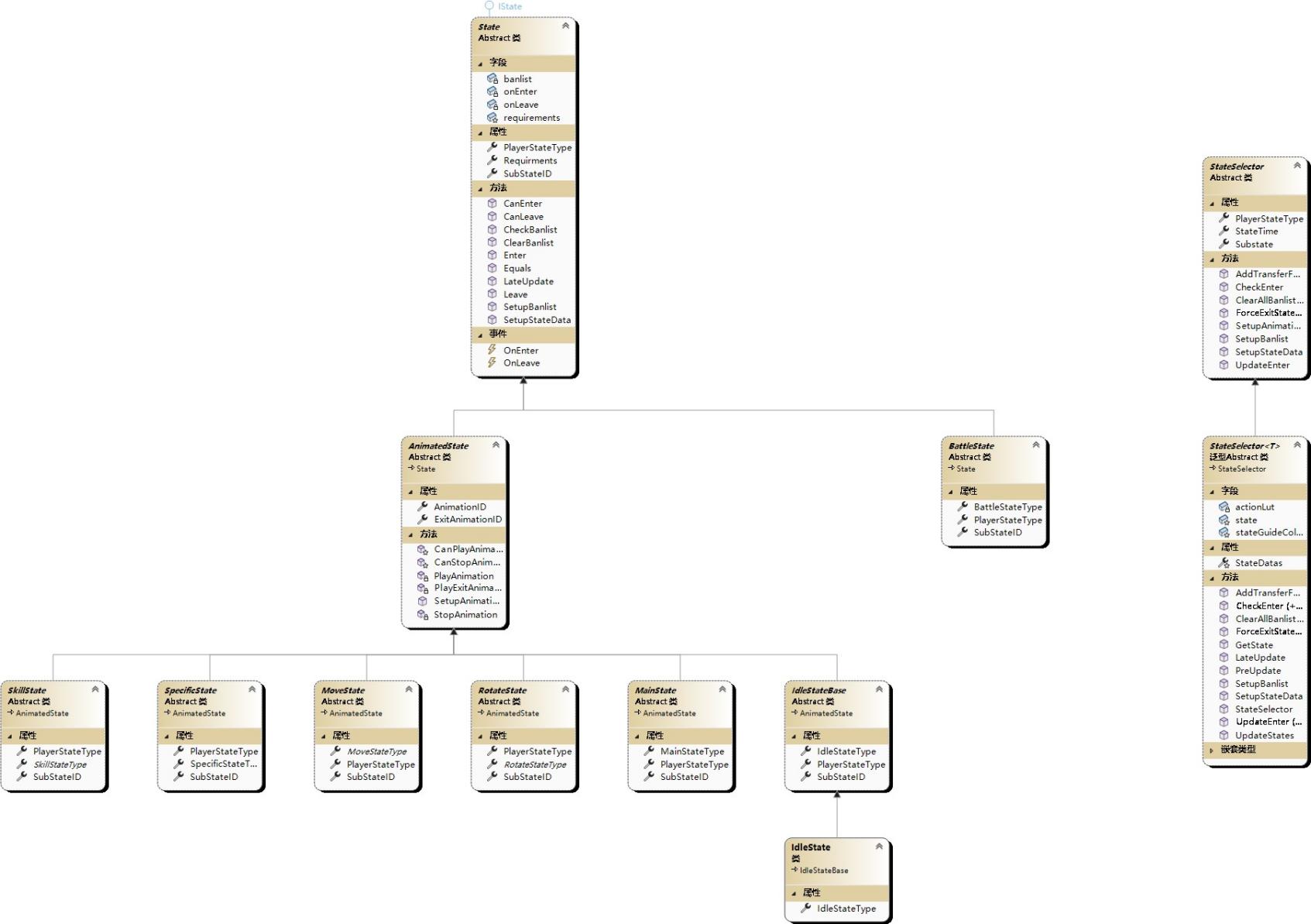


图 5-6 有限行为状态机类图



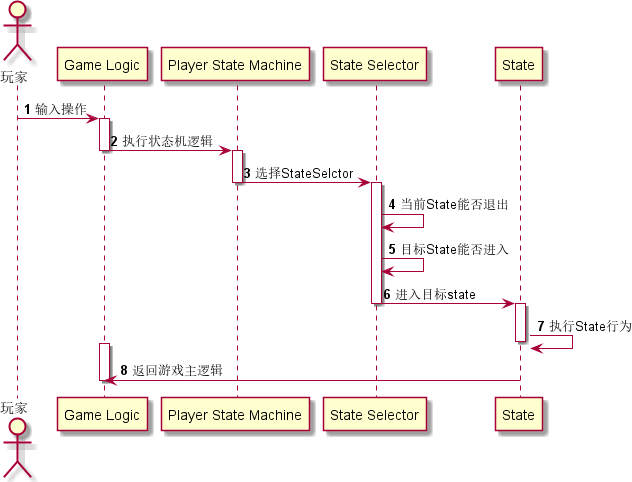


图 5-6 有限行为状态机时序图

* 1. 管理器模块设计

管理器模块主要负责游戏中各个逻辑模块的初始化和管理工作继承自LazyInstan ce类和接口IManager，LazyInstance类用来在调用时才会初始化该类型的单例，主要包括ExecutePrivioty标签。

ExecutePrivioty标签用来控制管理器的初始化顺序，在Game中会加载多个Manag er，为了保证Manager加载顺利，需要按照一定顺序加载各个Manager，使用ExecuteP rivioty从大到小来加载。

Managers模块时序图流程描述：1.游戏逻辑创建多个管理器，2.LazyInstance接口负责在调用时创建Manager单例，3.按照ExecutePririoty对Manager列表进行排序， 4.调用Init()方法初始化管理器资源，5.调用Start()方法执行开始游戏的逻辑，6.I update接口函数提供OnUpdate方法，7.执行OnUpdate方法，8.返回游戏主逻辑，9.St op()方法退出管理器，10.在Stop()之后执行Release()方法回收管理器资源。

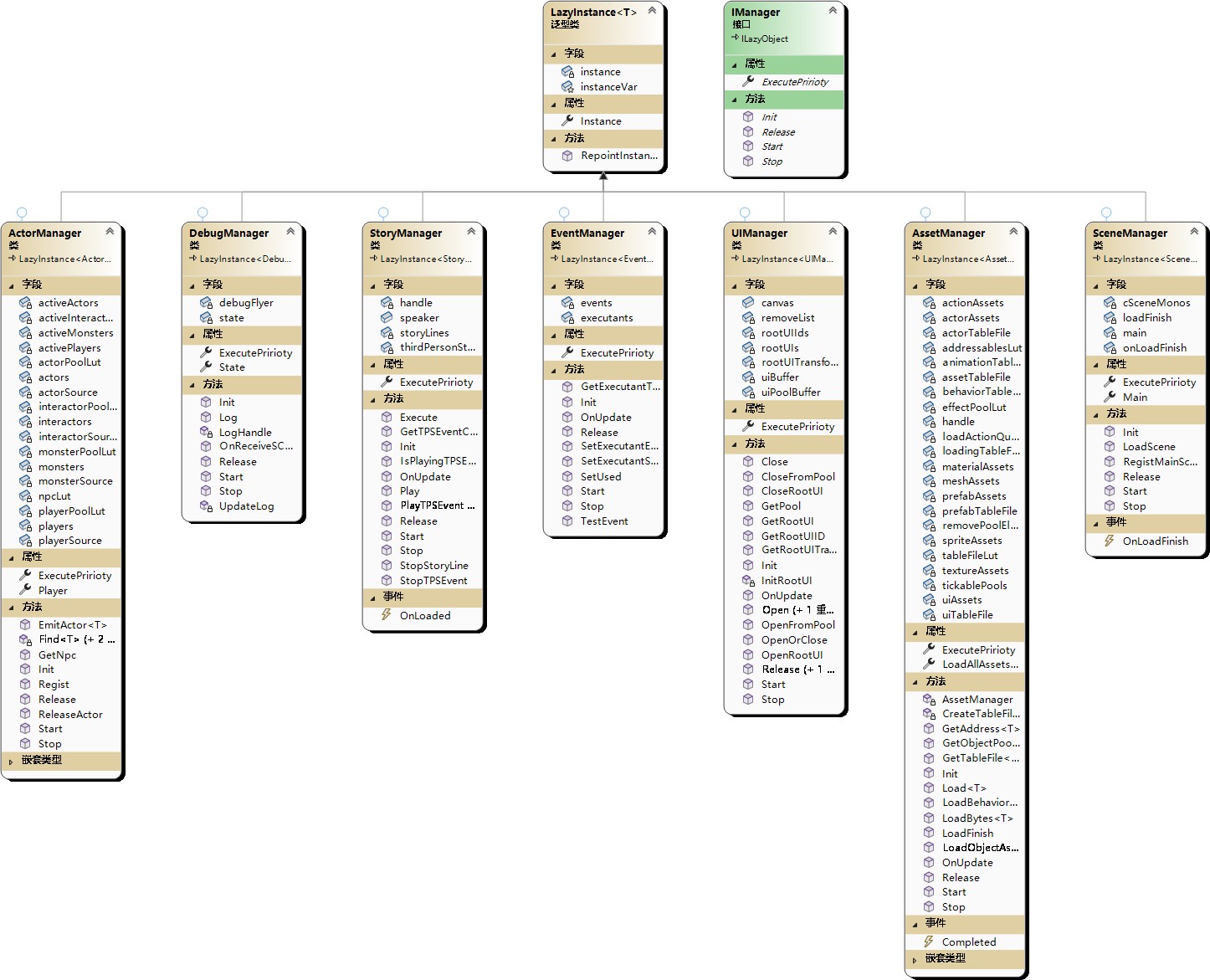


图 5-7 管理器模块类图

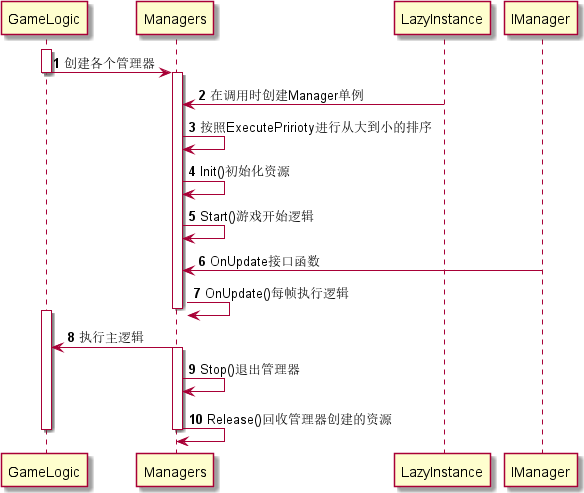


图 5-8 管理器时序图

* 1. 渲染模块设计

渲染模块控制游戏内多种多样的显示效果，实现不同逻辑的表现效果，从而丰富游戏的视觉表现。使用HLSL语言编写Shader，然后通过材质来给Shader更新数据，编

写脚本来控制材质的数据，同时来更新GPU中的数据，来实现实时更新渲染效果的功能。脚本主要基于RenderEffectController基类，包含mode字段和Pririoty、ActorRe

nderEffectType标签，其中：

mode字段保存当前效果是Once（播放一次）或者Loop（循环播放）。

Pririoty标签是一个只读标签，用来返回当前Effect的优先级，控制不同效果的先后顺序。

渲染模块时序流程图描述：1.RenderEffectController脚本向AssetManager中查询Material和Shader的数据，2.AssetManager返回查询到的数据，3.RenderEffectCon troller加载Shader资源，4.RenderEffectController加载Material资源，5.RenderEf fectController向Material中更新数据，6.Material向GPU中更新数据，7.GPU通知显示设备更新显示画面

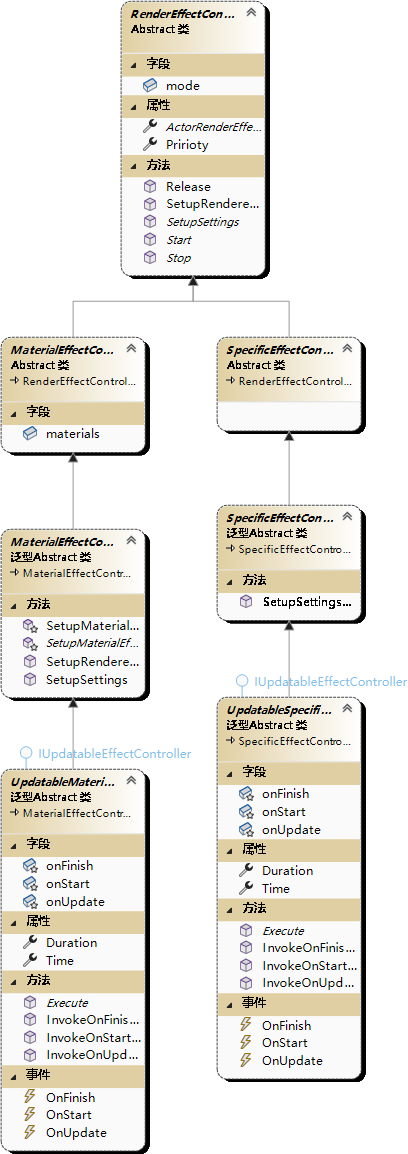


图 5-9 渲染功能模块图

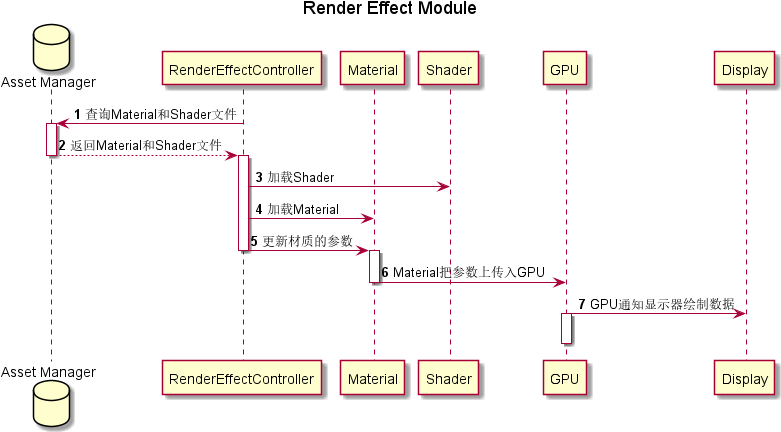


图 5-10 渲染功能时序图

* 1. 开始游戏模块

该模块负责玩家登录游戏的逻辑，进入游戏之后会初始化各个Manager，加载游戏资源。

该模块主要使用GameMain、FarmingGame，FarmingSystem类，GameMain类负责初始化游戏的各个模块，FarmingGame类负责种地游戏的相关资源初始化、主角的初始化和Fa rmingSystem的实例化，FarmingSystem类负责游戏的主逻辑和农作物生长等逻辑。

开始游戏模块时序图流程为：1.创建GameMain实例，2.调用Init()方法创建各个Ma nager实例，并初始化各个Manager，3.调用Start()方法来启动游戏，4.创建农场游戏实例，5.加载游戏场景，6.创建游戏主角实例，初始化主角，7.创建游戏主逻辑系统，8. 返回游戏主逻辑，9.通知玩家游戏加载完毕，并进入游戏

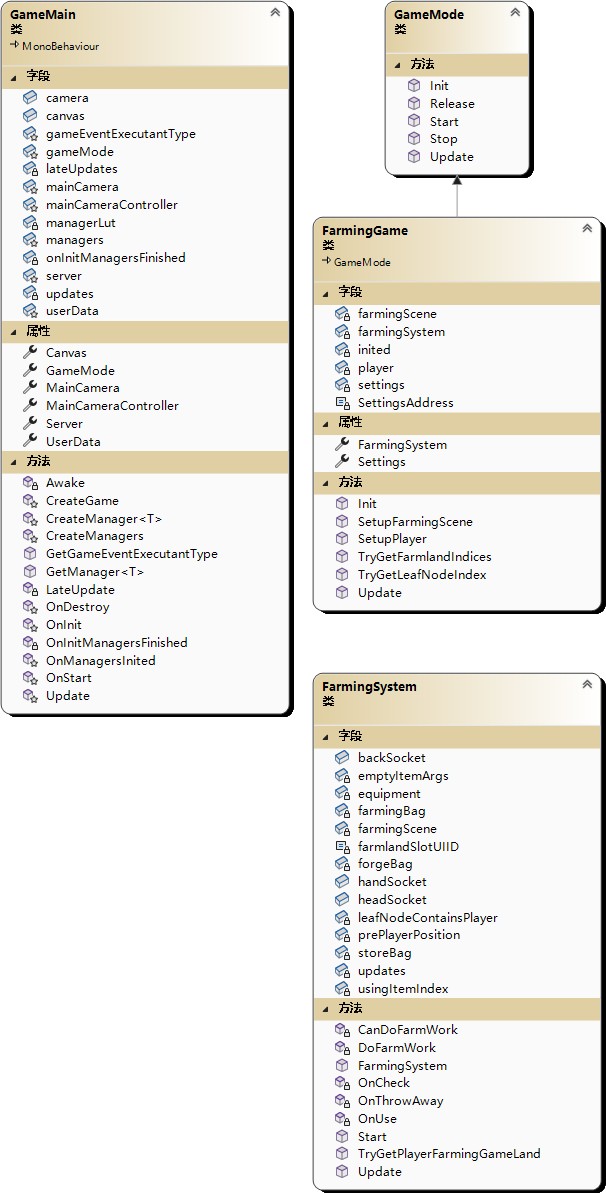


图 5-11 进入游戏模块类图

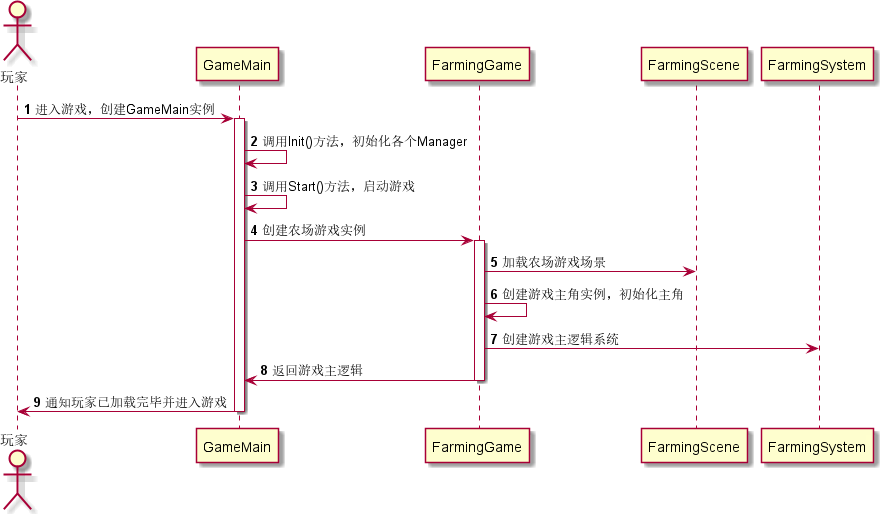


图 5-12 进入游戏模块时序图

* 1. 场景管理模块

该模块使用四叉树结构管理管理场景碰撞信息和可互动角色信息。

该模块主要包括FarmingScene类、QuadTreeFarmland结构体、FarmingGameLand 类，其中FarmingScene类主要用来生成并保存四叉树的场景信息，保存QuadTreeFar mland的叶子节点信息。QuadTreeFarmland用来保存叶子节点的地块信息。FarmingG ameLand是农场的子地块，我把一块农场分为N个子地块，再把每一个地块分为X\*Y个格子

场景模块时序图流程为：1.FarmingSystem加载游戏场景，2.初始化四叉树结构， 3.将场景栅格化,4.玩家输入游戏操作，5.查询玩家所在地块信息,6.递归四叉树结 构，查询玩家所在位置叶子节点信息，7.获取格子的碰撞以及格子物品等信息，8. 返回格子信息，9.反馈玩家操作

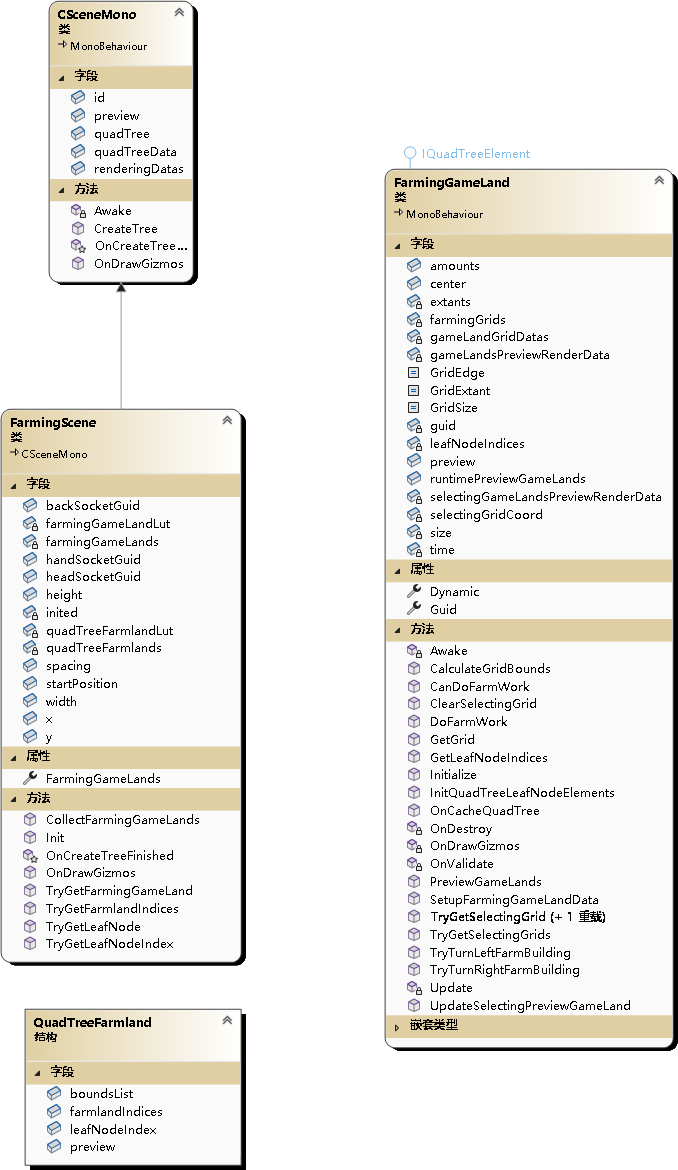


图 5-13 场景管理模块类图

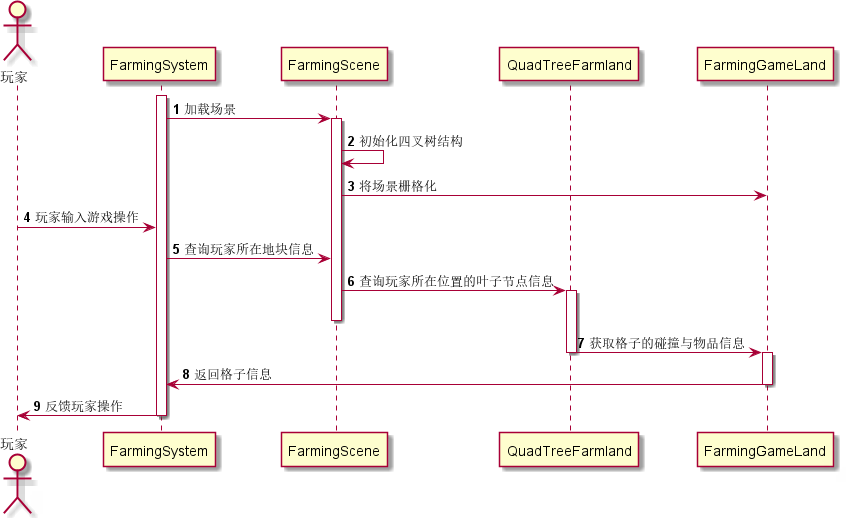


图 5-14 场景管理模块时序图

* 1. 背包模块

背包模块主要用来管理玩家的道具、金钱等，以及NPC的商店道具。

背包模块主要使用到Bag<T>,FarmingBag、FarmingItem、FarmingItem、Farming ItemEntry类。其中Bag类为背包的基类，派生出多种背包类。FarmingItem类为游戏道具类。FarmingItemEntry为道具词条类，用来实现道具的多种效果。

背包模块时序流程图为：1.FarmingSystem初始化背包实例，2.玩家输入打开背包操作，3.实现打开背包逻辑，4.玩家输入选择道具操作，5.实现选择道具逻辑，6. 玩家输入使用道具操作，7。实现使用道具逻辑，8.实现使用道具的多种效果，9.返回主逻辑，10.反馈使用道具

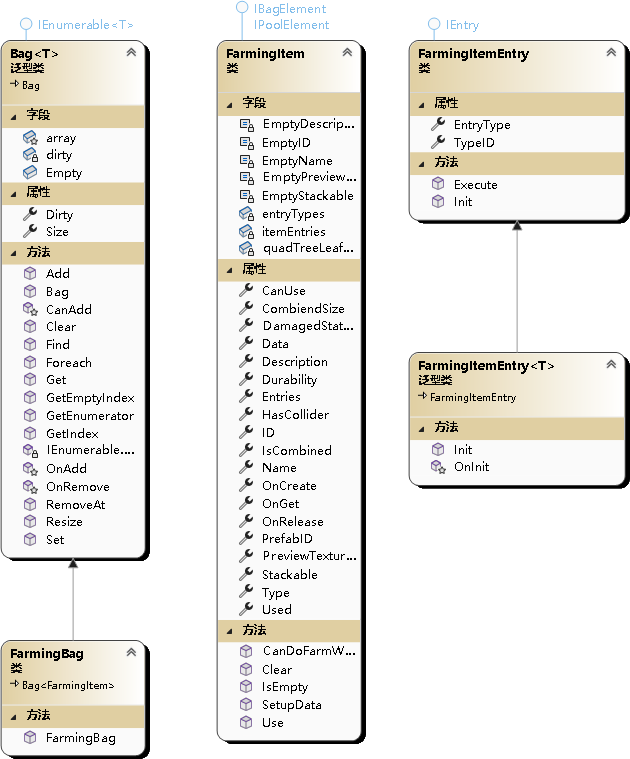
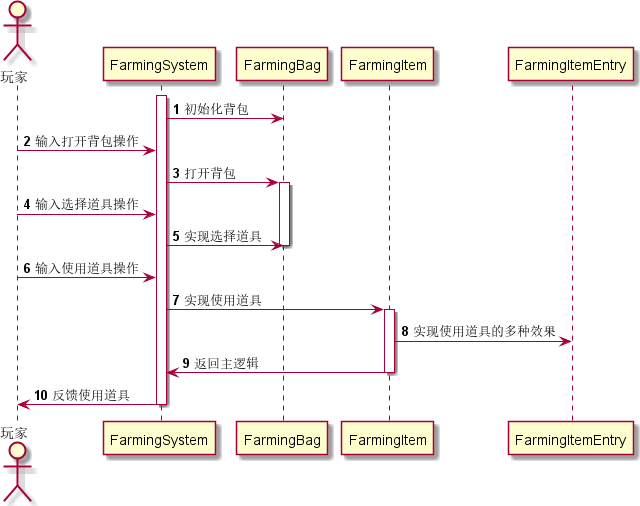


图 5-15 背包模块类图



5.16 背包模块时序图

* 1. 自定义农场模块

自定义农场模块用于玩家自定义农场单元，摆放不同的农场道具或者种植种子。

自定义农场模块主要使用FarmingGameLand，FarmingGrid，FarmBuilding三个类。其中FarmingGameLand负责管理场景中的自定义子农场块，FarmingGrid用来管

理单个格子，FarmBuilding用来实现格子内农作物或者农场建筑。

自定义农场模块时序流程图为：1.游戏主逻辑将农场栅格化为x\*y个格子，2.玩家选择需要编辑的地块，3.在背包中使用改变地块所需要的道具，4.玩家角色播放动作，5.玩家动作播放完毕通知游戏主逻辑，6.游戏主逻辑改变地块的类型，7.游戏主逻辑通知渲染模块改变地块外貌，8.完成改变地块外貌，9返回游戏主逻辑

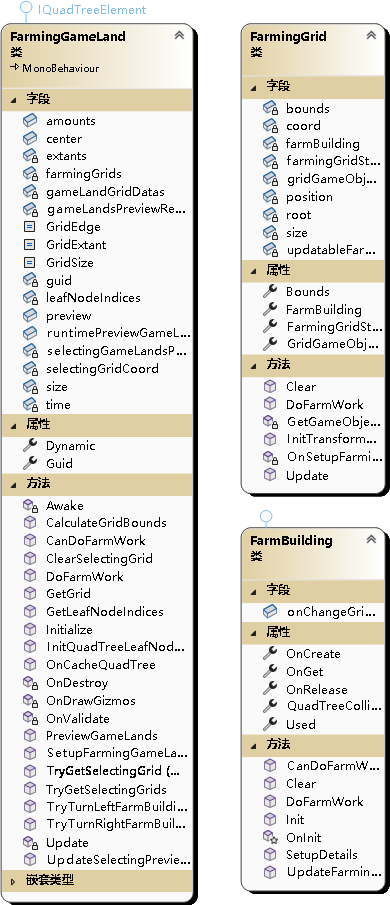


图 5-17 自定义农场类图

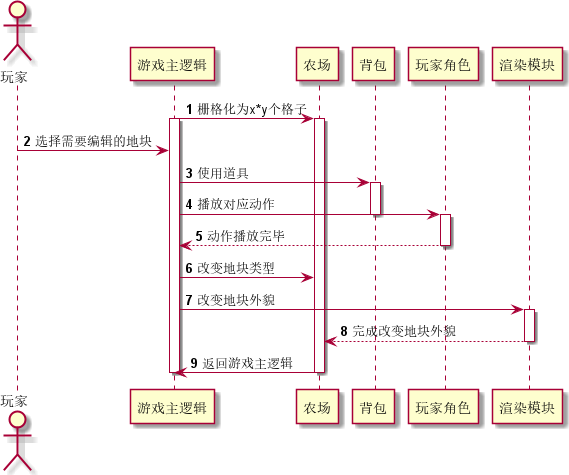


图 5-18 自定义农场模块时序图

* 1. 种植农作物模块

种植农作物模块用于实现玩家在目标地块种植农作物逻辑。

种植农作物主要使用到FarmingSystem、FarmingGameLand、RenderManager、UImanager类。其中FarmingSystem用来处理玩家输入，FarmingGameLand获取玩家所在地块信息，RenderManager处理渲染逻辑，UIManager负责管理UI界面。种植农作物模块时序流程图为：1.玩家选择目标地块，2.玩家打开背包选择种子，3. 玩家关闭背包，4.玩家向游戏主逻辑查询当前地块信息，5.游戏主逻辑向场景管理请求获取当前地块信息，6.场景管理返回当前地块信息，5.游戏主逻辑向场景管理请求获取当前地块信息，6.场景管理返回当前地块信息，7.游戏主逻辑向渲

染模块请求高亮当前模块，8.渲染模块执行高亮操作，9.游戏主逻辑向玩家展示当前模块，10.玩家向游戏主逻辑输入使用种子操作，11.主逻辑请求游戏角色播放种植动作，12.返回种植动作播放完毕，13.种植完成后通知地块改变状态，14.通知渲染模块改变地块外貌，15.渲染模块改变地块外貌，16.返回游戏主逻辑，17.反馈种植操作

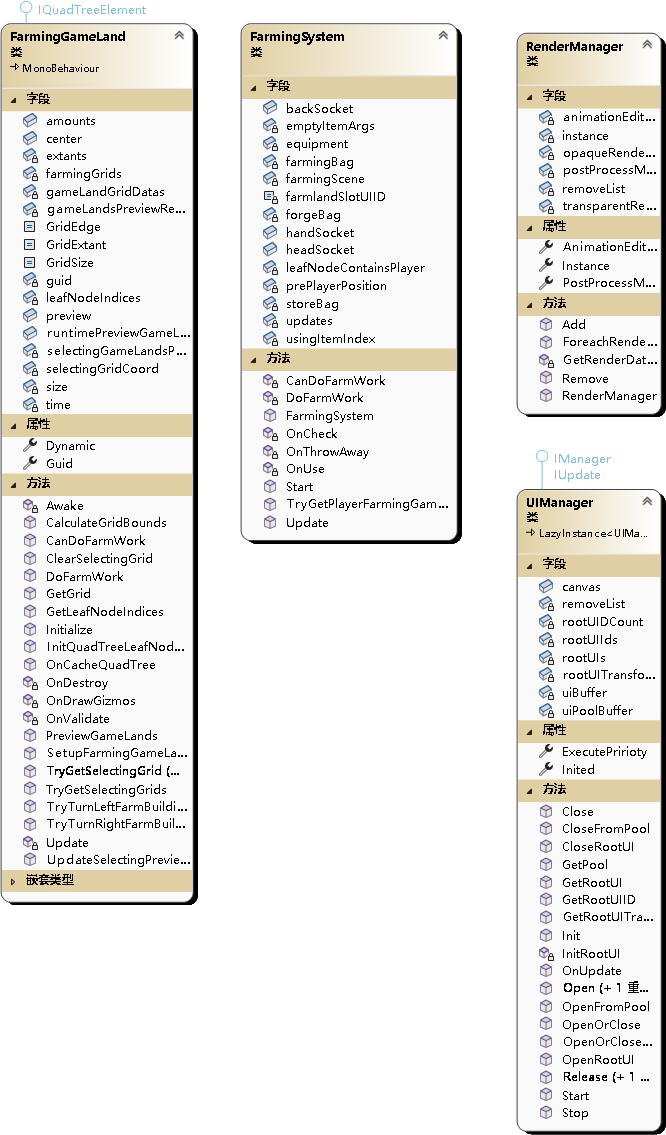


图 5-19 种植农作物模块类图



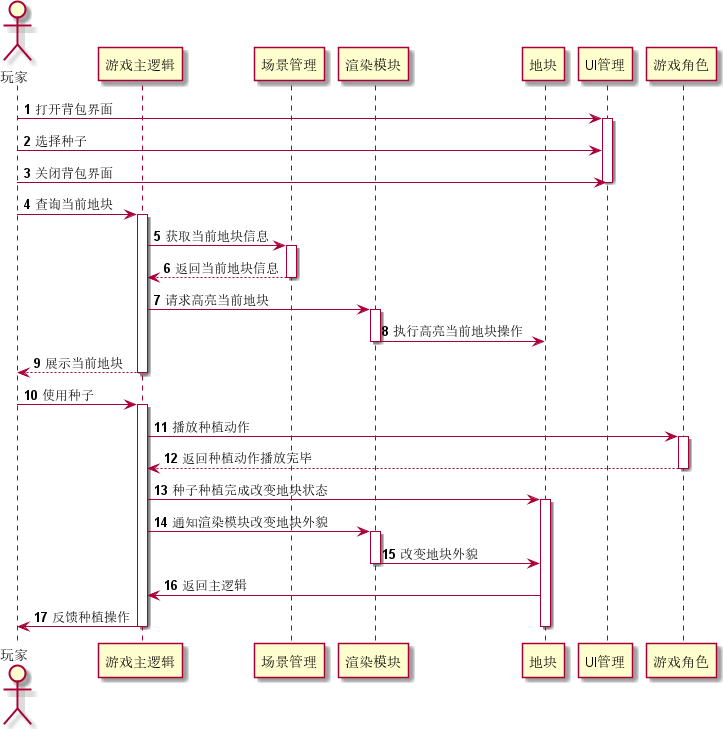


图 5-20 种植农作物模块时序图

* 1. 农作物生长模块

农作物生长模块用于实现农作物的生长逻辑。

农作物生长模块主要使用Farmland、FaringObject、CropsObject类。其中Farmlan d用来管理当前格子，FarmingObject为格子内道具的基类，CropsObject为格子内的农作物类。

农作物生长模块时序流程图为：1.游戏主逻辑请求天气系统查询当时天气情况，2. 返回天气情况，3.确认晴天状态，4.增加农作物渴水值，5.请求确认农作物渴水值超 过阈值，6.返回是否超过阈值，7.请求地块改变状态为干涸，8.地块改变自身状态为 干涸，9.游戏主逻辑请求渲染模块改变地块地貌，10.渲染模块改变地块外貌，11.确 认下雨状态，12.增加农作物需水值，13.请求确认农作物需水值超过阈值，14.返回是否超过阈值，15.请求地块改变状态为干涸，16.地块改变自身状态为干涸，17.游戏主逻辑请求渲染模块改变地块地貌，18.渲染模块改变地块外貌

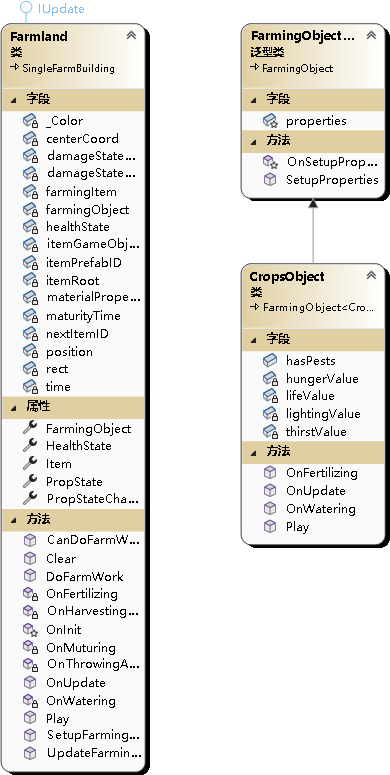
图 5-21 农作物生长模块类图



图 5-18 农作物生长模块时序图

* 1. 序列化与反序列化模块

序列化与反序列化主要使用C#中的BinaryFormatter进行二进制的序列化，以及U nity的Addressables序列化，使用了Serialization静态类。主要是将游戏内数据转换为字节流，然后使用BinaryFormatter.Serialize()方法进行本地序列化，序列化后的文件可以使用BinaryFormatter.Deserialize()方法反序列化，生成资源。

序列化时序图流程为：1.将需要序列化的数据传入Serialization，2.使用Seria lization将数据转换成byte数组，3.释放中间资源，4.返回byte数

组，5.将byte数组写入二进制文件

反序列化时序图流程为：1.加载序列化好的文件，2.返回加载好的序列化文件， 3.读取数据写入byte数组，4.使用Serialization将byte数组转换成类或者结构体实例，5.返回实例，6.释放byte数组

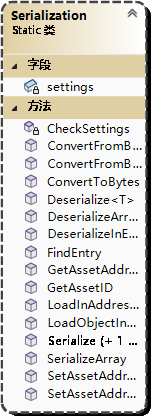


图 5-19 序列化和反序列化类图

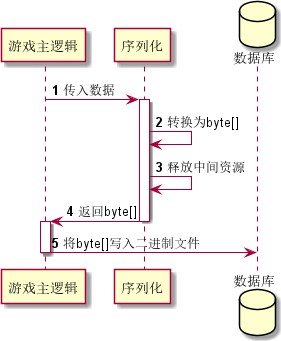
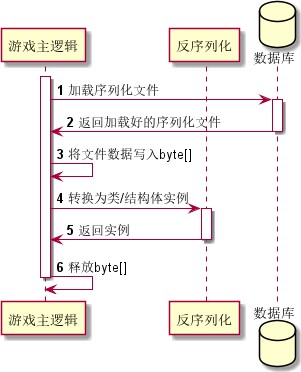
 

图 5-20 序列化时序图流程 图 5-20 反序列化时序图流程

* 1. 数据库设计

本系统使用自定义编辑器，使用编写好的表结构，存放着游戏中预先编辑好的数 据，使用二进制序列成本地文件，数据供需 5 张表。分别是角色信息表、道具信息表、对话剧情表、资源信息表、场景管理表。

* + 1. 角色信息表

用户信息表具有角色ID，角色名称、路径、guid、金钱、移动速度等信息。角色信息表属性图如下所示：

表 5-1 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 数据长度 | 允许空值 | 说明 |
| id | int | 4 | 否 | 角色id |
| name | string | 50 | 否 | 角色名称 |
| address | string | 50 | 否 | 角色资源路径 |
| guid | string | 20 | 否 | 全局唯一标识 |
| gold | uint | 4 | 否 | 金钱数 |
| moveSpeed | float | 4 | 否 | 移动速度 |

* + 1. 道具信息表

道具信息表具有道具ID、道具名称、道具价格、道具图片id、道具模型id

信息。道具信息表属性图如下所示：

表 5-2 道具信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 数据长度 | 允许空值 | 说明 |
| id | int | 4 | 否 | 道具id |
| name | string | 50 | 否 | 道具名称 |
| price | uint | 4 | 否 | 道具价格 |
| textureId | int | 4 | 否 | 图片id |
| meshId | int | 4 | 否 | 模型id |

* + 1. 对话剧情表

对话剧情表具有对话id、对话名称、npcId、对话内容信息。

对话剧情表属性图如下所示：

表 5-3 笔记信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 数据长度 | 允许空值 | 说明 |
| id | int | 4 | 否 | 对话剧情id |
| name | string | 50 | 否 | 对话剧情名称 |
| npcId | int | 4 | 否 | 对话NpcId |
| content | string | 500 | 否 | 对话内容 |

* + 1. 资源信息表

资源信息表具有资源id、资源名称、资源路径、guid、资源类型信息。资源信息表属性图如下所示：

表 5-4 订单信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 数据长度 | 允许空值 | 说明 |
| id | int | 4 | 否 | 资源id |
| name | string | 50 | 否 | 资源名称 |
| address | string | 50 | 否 | 资源路径 |
| guid | string | 20 | 否 | 全局唯一标识 |
| type | int | 4 | 否 | 资源类型 |

* + 1. 场景管理表

场景信息表具有起始坐标、尺寸、guid、地块类型、地块道具ID。场景信息表属性图如下所示：

表 5-4 场景信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 数据长度 | 允许空值 | 说明 |
| coord | Vector2Int | 8 | 否 | 起始坐标 |
| size | Vector2Int | 8 | 否 | 长宽尺寸 |
| guid | string | 20 | 否 | 全局唯一标识 |
| farmlandType | int | 4 | 否 | 地块类型 |
| itemId | int | 4 | 否 | 道具Id |

5.12 本章小结

本章针对模拟农场游戏系统中各个功能模块的设计做了详细的描述。Editor端模块主要有自定义表编辑器模块。Runtime端主要有行为树逻辑模块、有限行为状态机模块、管理器模块、渲染模块、开始游戏模块、场景管理模块、背包模块、自定义农场模块、种植农作物模块、农作物生长模块、序列化与反序列化模块。数据库使用自

定 义数据结构，以及自定义编辑器进行编辑，主要包含角色信息表、道具信息表、对

话信息表、资源信息表以及场景信息表共五张表，并对表的结构进行了详细说明。

1. 系统实现

模拟农场游戏是一个从编辑器到美术资源到游戏逻辑都需要开发的完整系统，研发人员制作编辑器并配置好表数据，玩家进入游戏、编辑农场、种植农作物、保护农作物生长、收获农作物、卖出等功能。

* 1. 自定义编辑器功能

自定义编辑器是指使用C#语言，自行抽象出的编辑器界面框架。将文本、数据、按钮、列表等功能抽象为函数，然后调整好页面布局，制作出所需要的编辑器界面。制作的表编辑器界面如下

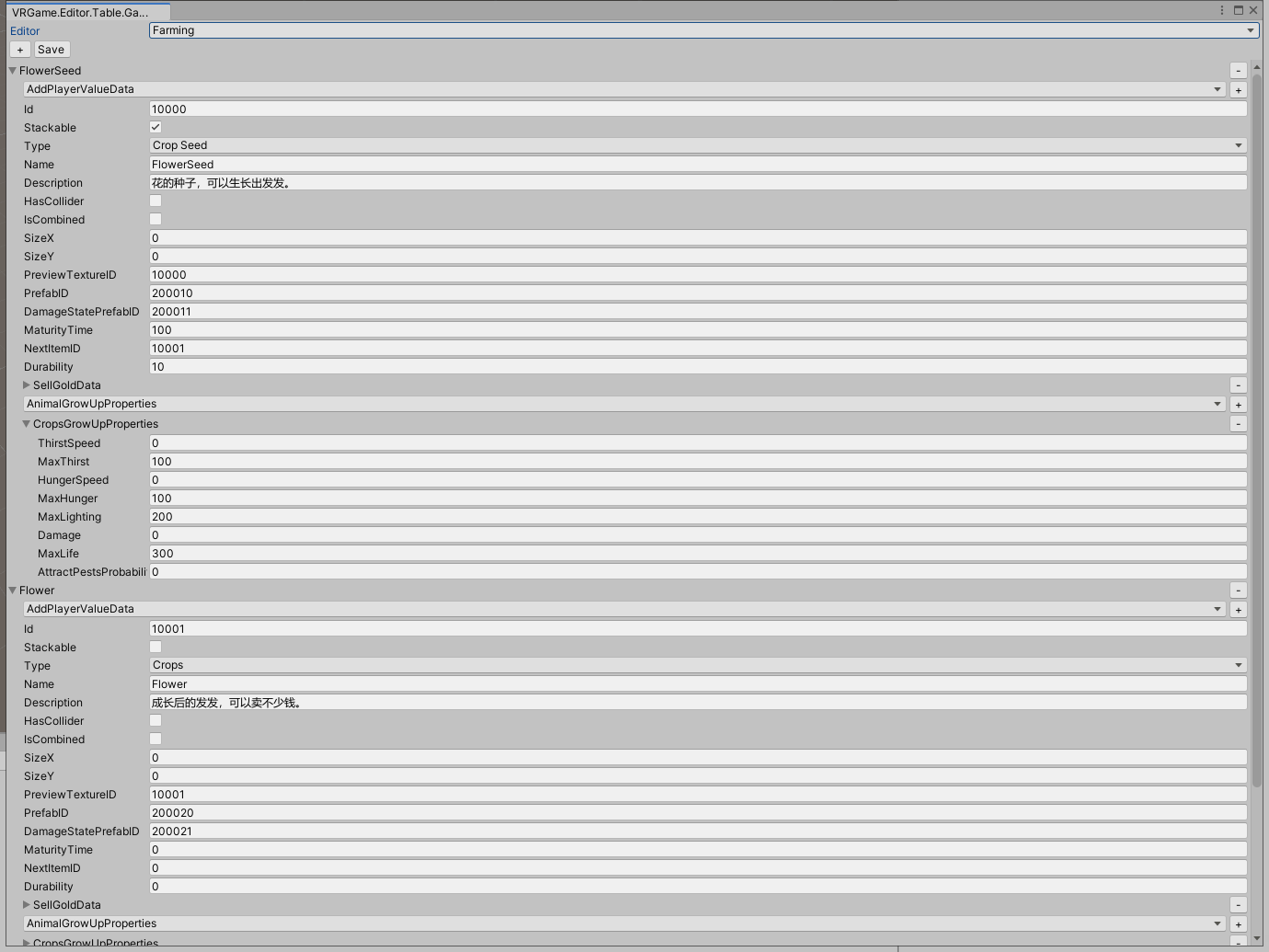


图 6-1 农场道具表编辑器界面

编辑器功能逻辑描述：

* + 1. 将各种控件抽象为多个方法
    2. 使用Vector2来存储控件的坐标和尺寸，控制控件的布局
    3. 将编辑器生成的数据序列化为二进制文件
    4. 在系统中反序列化二进制文件，生成需要的数据实例
    5. 该系统为通用系统，行为树的编辑器也使用该框架进行制作。
  1. 行为树模块

行为树系统主要负责实现NPC的各种逻辑，行为树系统使用树形结构，将逻辑抽象为节点形式，从根节点自上而下按照一定条件去执行。行为树节点的结构一般为当前节点所需要的数据以及下一个节点的guid，使用guid来保存下一个节点的位置，如果上级节点可以执行，那么该节点的子节点如果满足执行条件，会继续向下执行，直至无法执行或者到达叶子节点。

行为树节点可以分为根节点、条件节点、控制节点、行为节点四大类。根节点负责行为逻辑的开始调用工作。

行为节点为叶子节点，存在于树枝的末端，运行行为节点则表示该枝逻辑顺利执行完成。

条件节点和控制节点作为非叶子节点连接在根节点和叶子结点之间。条件节点用来判断当前角色是否满足该条件节点，如果满足会执行条件节点的子节点，条件节点继承自ConditionNode，需要通过逻辑扩展多个条件节点，满足游戏中各种各样的逻辑需求。

控制节点用来控制逻辑的运行顺序，大致可分为SequenceNode、SelectorNode、ParallelNode三大类，控制节点一般会有多个孩子节点。SequenceNode会按照孩子节点的顺序依次调用。SelectorNode会按照孩子节点的顺序尝试调用，如果一个孩子所在的树枝顺利执行完成，那么会返回不再继续执行。ParallelNode会并行的同时执行所有的孩子节点，直到所有的孩子节点都执行完毕。

控制节点和条件节点没有先后顺序，但是都必须在行为节点之前，行为节点不可以添加子节点。

行为树的编辑器系统基于自定义编辑器功能进行绘制，编辑器生成的行为树资源会被序列化为二进制文件，跟角色加载时一起加载，并生成该角色的逻辑实例。

此外行为树编辑器实现了添加节点、自动排序、合并为子行为树、行为树数据所需要数据的编辑器等功能。大大提高了编辑速度，可以很方便的实现出游戏中所需要的各种逻辑。

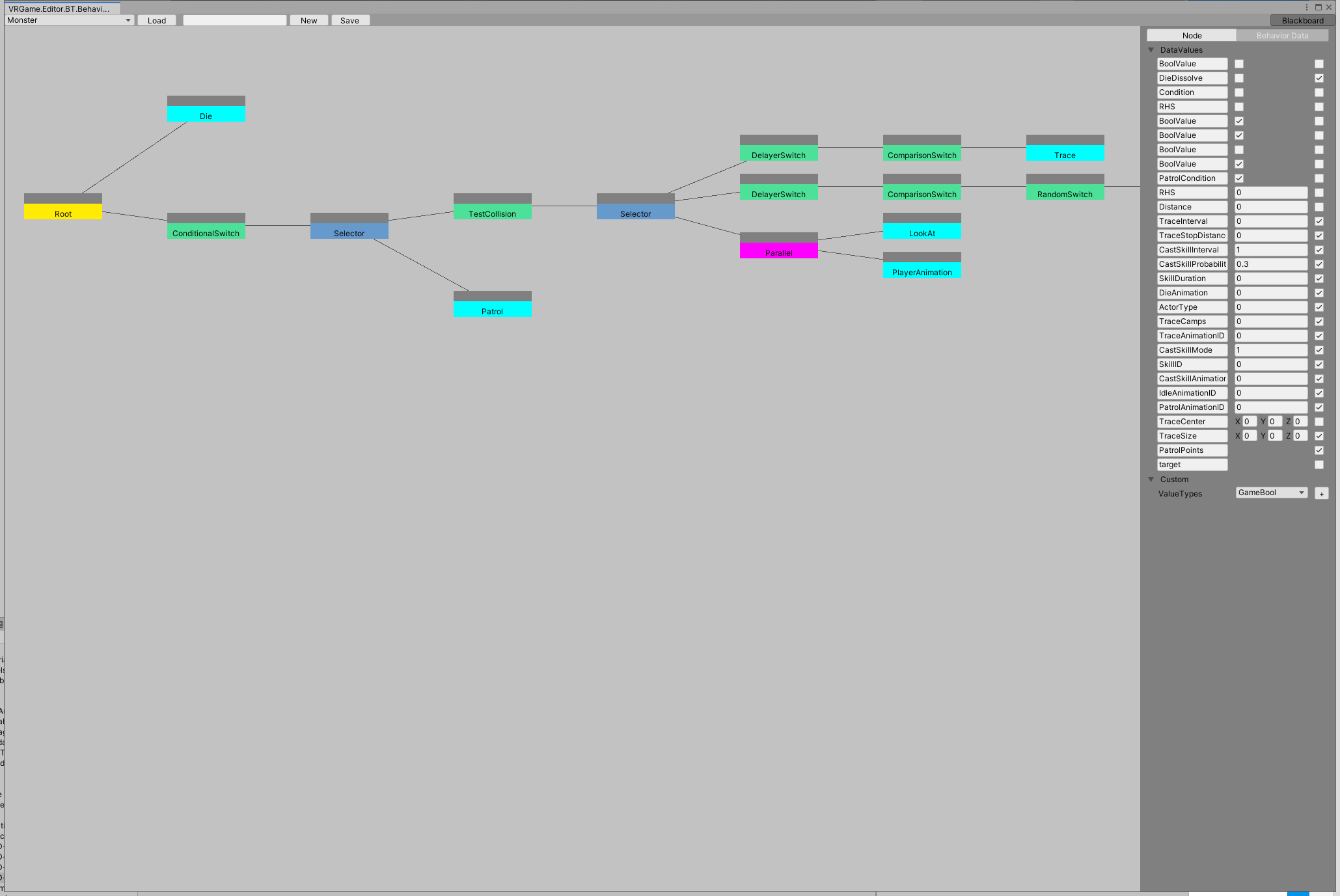


图 6-2 行为树编辑器功能1-数据编辑



图 6-3 行为树编辑器功能2-添加节点

编辑完成的行为树文件，可以在角色的脚本上检索到，然后行为树的数据会显示在脚本编辑器上。通过编辑角色上的行为树数据，可以实现对于同一个行为树逻辑，应用在不同的角色之上，编辑不同的数据也可以生成不同的逻辑行为，例如指定不同的动作ID，可以播放不同的动作；指定不同的UIID，可以打开不同的UI界面。通过这样的设计，可以二次扩展逻辑的通用性，进一步提升工作效率。



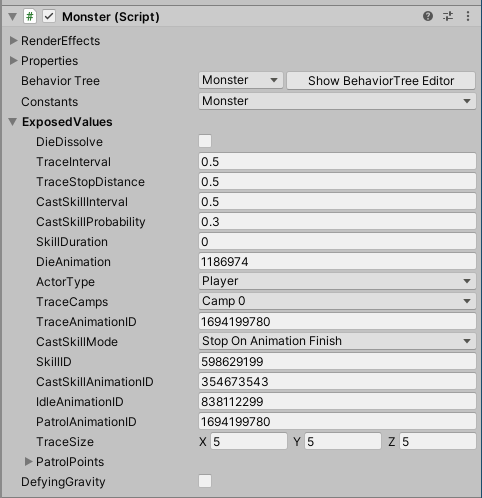


图 6-4 角色脚本上的行为树数据编辑功能

行为树功能逻辑描述：

1. 编写行为树编辑器功能
2. 使用行为树编辑器生成行为树逻辑文件，序列化为二进制文件
3. 在角色脚本上选择不同的行为树逻辑，并编辑该逻辑的数据
4. 在运行时加载角色资源，生成角色脚本实例，然后反序列化行为树资源 文件，生成行为树逻辑实例
5. 运行行为树逻辑
   1. 有限行为状态机模块

有限行为状态机模块是用来实现主角的各种逻辑。与行为树不同的是，主角的逻辑是单一且确定的，并且主角的逻辑是复杂的，如果使用行为树去制作会非常复杂， 且难以维护，所以将主角逻辑拆分开来单独制作。

有限行为状态机（FSM）将主角分为有限多个状态，包括待机状态、主状态、混合状态。其中：

待机状态表示角色没有做任何行为，处于待机状态，播放待机动作等待玩家指令。主状态表示主角当前所处的主要状态，该状态为唯一状态，包括站立状态、跳跃

状态、下蹲状态等。

混合状态表示主角的行为状态，该状态由多个状态混合而成，混合状态包括移动状态、旋转状态、技能状态。这三种状态之间可以同时存在，表示玩家可以边移动边旋转，或者边移动边旋转边释放技能。但是这三种状态也有唯一情况，例如玩家向前移动和向后移动不能同时存在；玩家释放一个技能的同时不能释放另外一个技能。通过状态混合可以完成角色的各种逻辑。

此外状态之间需要相互转换，例如从待机状态进入移动状态，通过定义StateSel ector类来处理状态之间的转换工作。该类的工作流程为，1.判断是否要进入新的状态，2.判断当前状态和目标状态不是同一状态，3.判断能否退出当前状态，4.判断能否进入目标状态，5.执行退出当前状态的逻辑，6.执行进入目标状态逻辑。

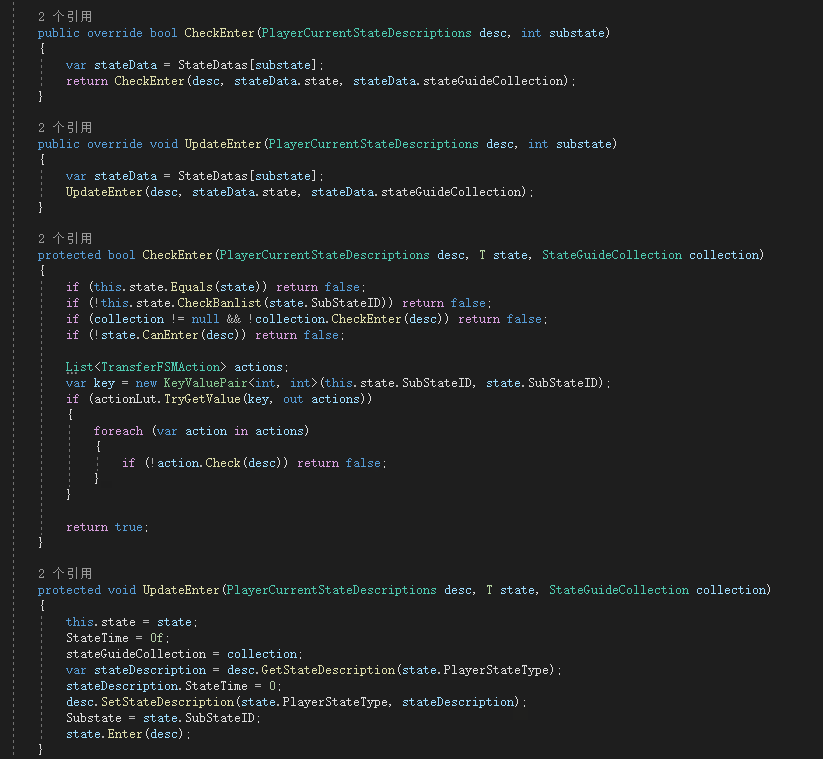


图 6-5 目标状态的判断和状态的转换



图 6-6 行为状态的进入和退出逻辑

FSM功能逻辑描述：

1. 主角实例化后，默认进入待机状态
2. 判断是否有玩家的输入操作，以及是何种操作
3. 根据输入操作寻找需要进入的状态的类型
4. 判断当前状态能否退出以及目标状态能否进入，执行退出逻辑和进入目标状态逻辑
   1. 管理器模块

管理器模块负责管理游戏内多种逻辑，是较为底层的游戏逻辑，主要包括资源管理器、天气管理器、场景管理器、角色管理器、UI管理器等。其中：

资源管理器负责管理资源的加载和卸载，包括贴图、模型、预制体等资源。

天气管理器负责管理游戏中动态天气系统，实现了24小时天气的变化，从早到晚的时间流逝以及下雨、晴天等天气的动态管理。

场景管理器负责动态加载、卸载场景，以及使用四叉树结构管理场景内物件，方便逻辑高效处理查找、选择物件的逻辑。

角色管理器负责管理角色的加载、卸载，使用池系统来管理角色。由于C#语言的特性，C#会使用自带的GC来管理内存，在方便的同时也会带来极大的性能消耗，触发G C时会造成巨大卡顿。为了防止游戏频繁触发GC，使用池来管理托管资源是一种高效的操作。可以被池管理的类继承自IPoolElement接口，该接口实现了Used标签，当池中的第一个物体Used为true或者该池数量为0是会创建实例，使用完毕会将Used设为fals e，并将该实例置于列表头部。如果第一个实例Used为false，那么会取出该实例，将U sed设为true，然后置于列表尾部。通过这样的操作，只需要判断第一个实例即可，整个操作是一个非常高效的操作。

UI管理器负责加载、卸载UI界面，该管理器含有一个枚举变量，通过该枚举可以管理一些单例UI，不需要使用该UIID即可实现加载、卸载操作。

管理器模块通常与表模块紧密结合，将编辑好的表数据加载出来供逻辑使用，其他游戏逻辑都统一从管理器模块中拿到需要的数据或者创建需要的实例。

* 1. 美术资源制作

前面介绍了程序底层的一些设计，实现了数据库的制作、存储和加载，角色逻辑的底层原理，以及几个底层逻辑管理器的实现。除了程序方面需要做的工作以外，还需要制作美术所需要的资源，其中最主要的包括模型、贴图、动作资源。这些资源的制作需要使用到专门的美术工具去制作。

模型资源可以使用3DMax、Maya、Blender等多种工具去制作，这里使用最常用的的工具3DMax去制作，模型资源主要提供了网格的顶点信息，包括顶点坐标、法线、UV 等，顶点坐标可以用来计算模型的世界坐标信息，显示在正确的位置，法线用来计算光照，UV则用来显示贴图资源。

贴图资源一般使用Photoshop工具进行制作，对于模型的贴图，需要配合模型的U V去绘制。

动作资源也使用3DMax去制作，可以和角色模型一并制作。

制作完成的上述资源在导入到引擎之后便可以使用在游戏中，通过在表编辑器中编辑好资源的ID、路径等信息，就可以在游戏中动态的加载这些资源。资源的截图如下：



图 6-7 角 色 模 型 图 6-8 角 色 贴 图

* 1. 编辑游戏场景

准备好需要的美术资源文件之后，可以开始编辑游戏的场景，编辑游戏场景需要在引擎内使用引擎的编辑工具进行编辑。

该农场游戏使用的是锁定视角，且游戏单位处于同一高度平面上，将制作好的地面模型拖入游戏场景中，放置在起始点（0,0,0）的坐标位置上，在后续的流程中需

要将地面栅格化，地面的坐标对齐有利用后续栅格化的计算。拖入地面后的场景如图：

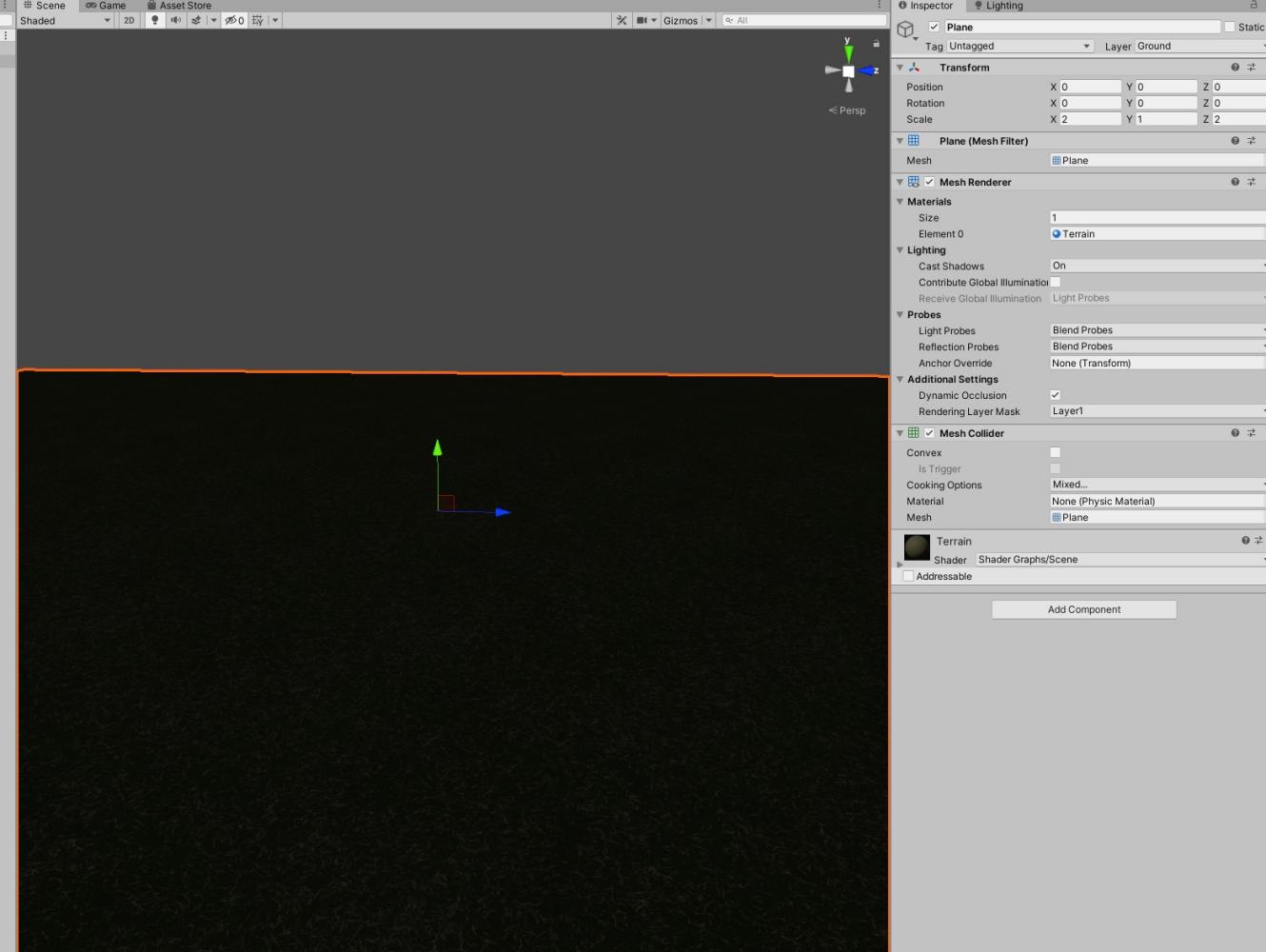


图 6-9 没有光照的地面

拖入地面后，因为场景中没有灯光，所以地面是黑的，我们需要添加一盏方向光来模拟太阳光的光照，拖入灯光后的场景如图：

之后我们需要在场景中添加天空球材质，这样在游戏中渲染出实时变化的天空球效果，同时由于游戏采用PBR光照的原因，天空球的光照也会作为GI对地面的光照产生影响，拖入天空球后的场景如图：

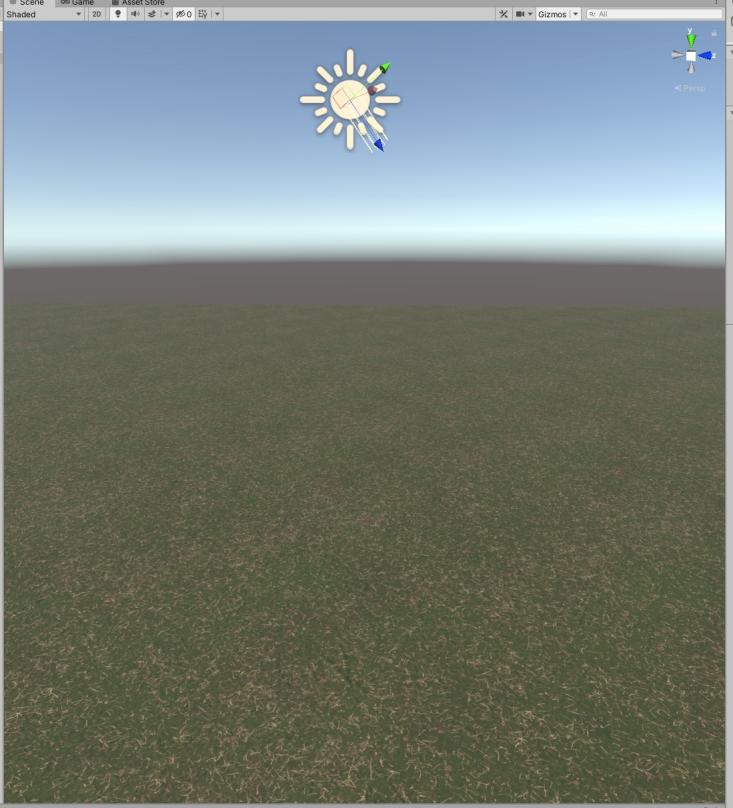
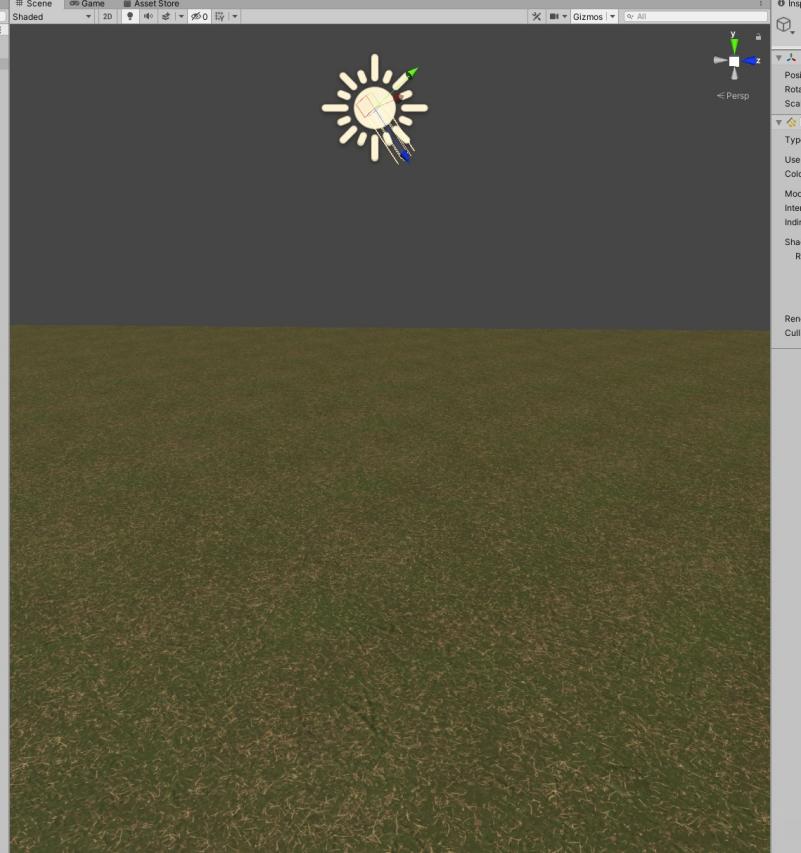


图 6-10 拥有光照的地面 图 6-11 拥有光照和天空球的地面

编辑完成场景之后，将场景文件保存在本地，然后添加在场景表中，之后可以使用场景管理器加载该场景。

* 1. 物理碰撞实现与场景管理

物理功能是为了实现物体与物体之间的真实的物理效果，可以实现角色与地面的碰撞、角色与其他NPC之间的碰撞等。这里仅仅使用包围盒实现了物体之间的碰撞检测功能，通过检测到物体之间是否碰撞来实现角色站立在地面上，以及栅栏、房屋等的碰撞效果。

碰撞检测可以基于多种检测方法，使用包围盒对包围盒是最为简单的一种，并且效率非常游戏，非常适合轻量级游戏去使用，其他还有三角面对三角面、三角面对包围盒等碰撞检测方式。包围盒有两种一种是有向包围盒（Oriented-Bounding-Box，简称OBB）和轴对齐包围盒（Axis-Aligned-Bounding-Box，简称AABB）。OBB是可以旋转的包围盒，AABB是和坐标系对齐的包围盒，AABB计算简单，后续的计算都是基于AABB 进行计算的。

AABB包含4个字段，min，max，center，size，其中：min表示AABB最小的坐标点， max表示AABB最大的坐标点，center表示AABB的中心点，size表示AABB的尺寸。

AABB与AABB的碰撞非常简单，这里用aabb1和aabb2表示两个AABB，我们只需要判断两个AABB中心点坐标的差小于两个包围盒尺寸之和即可。

当角色移动时，判断角色的AABB与其他物体的AABB是否碰撞，即可判断出是否有物体会阻挡角色移动。

给场景中每个物体一个合适的AABB，之后需要使用四叉树结构来预处理场景，将 场景中物体的AABB信息整理成四叉树数据结构。四叉树结构是一种树形数据结构，可 以用来大幅加速场景中物体遍历效率。类似的结构还有八叉树结构或者BVH（Bounding- Volume-Hierachy），其中四叉树实现简单，且模拟农场游戏都在同一高度平面，不需要使用八叉树结构。

四叉树结构的原理是将场景中所有物体的AABB合并为一个AABB，然后在x、z两个方向上将AABB等分成四个AABB作为上个AABB的子集，重复该操作直至叶子AABB的尺寸达到某个阈值，即最小尺寸不能超过一定值，然后子节点收集和场景中物体的AABB的相交情况，如果相交则保存在该节点下。在遍历时先从最根部的节点去检测根节点AAB B是否与目标AABB相交，如果相交则判断子节点是否相交，如果相交则继续向下判断直至达到叶子节点，然后取出叶子节点的物体信息，则该物体与目标相交。使用这样的方法可以使原先遍历所有物体的操作变更为遍历n次的操作，设根节点AABB尺寸为x， 最小尺寸为y,则n = log2(x / y)次。



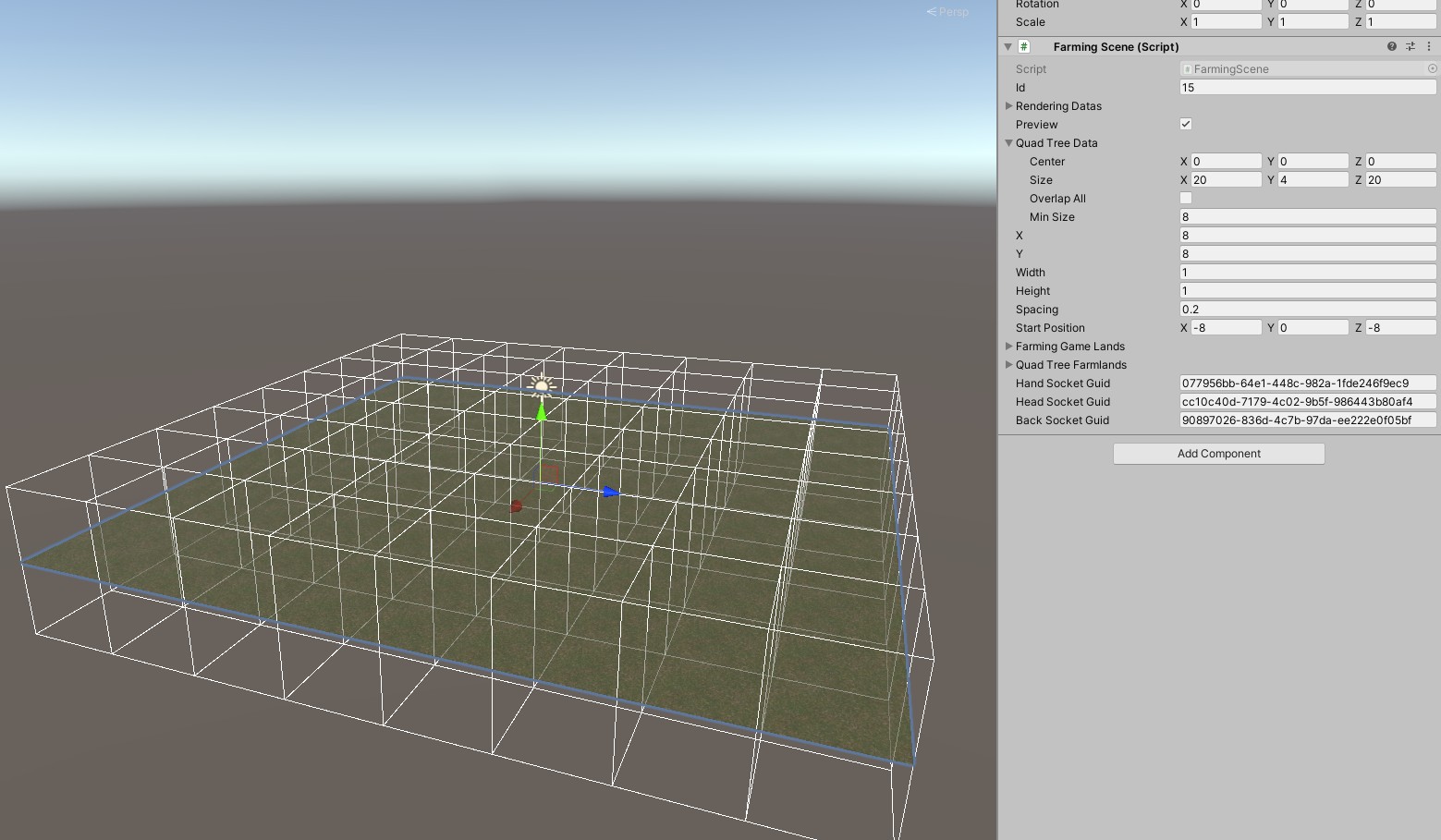


图 6-12 四叉树分割后的叶子节点

* 1. 开始游戏

经过上面一步一步的处理，我们已经做好了游戏资源的编辑工作，现在可以进入游戏了。进入游戏需要按照顺序进行初始化，首先要初始化好资源管理器，其他管理器的初始化可能会依赖于资源管理器，所以有限初始化资源管理器，Unity加载资源主要有同步加载和异步加载两种方式，同步加载资源使用主线程去加载资源，会导致资源加载完毕之前主线程卡死的情况，所以这里使用异步加载的方式去加载资源，即使用多线程去加载，当加载完毕会使用回调函数来通知加载完成，当个我们所需要的资源全部加载完成之后，使用回调去加载其他管理器。



当管理器全部加载完毕之后需要使用场景管理器加载游戏场景，然后使用角色管理器加载玩家角色和其他NPC，再使用UI管理器去加载需要UI界面。此时开始游戏需要的工作全部完毕，可以进入游戏了。



图 6-13 加载完毕，开始游戏



* 1. 天气系统

天气系统主要用来控制下雨天和晴天等天气的变化和出现情况，当下雨天出现时， 地理的农作物更容易出现过度灌溉的情况，影响农作物生长；晴天时阳光充足，农作 物生长的更快更健康。

天气包含晴天、阴天、小雨、大雨四个状态，用一个枚举来表示当前天气情况， 天气的变化需要一个均匀的变化过程，即从晴天变化到大雨需要从晴天变化到阴天、阴天到小雨、再从小雨到大雨这样一个变化过程， 我们可以把该过程映射成一个0-1 的函数去表示，然后用一些锚点值来表示天气的状态。在这里我规定0.65-1是晴天， 0.5-0.65是阴天，0.35-0.5是小雨，0-0.35是大雨，这样只需要一个0-1范围内的数值变可以表示当时的天气。这里我使用了噪波来生成一系列连续的随机数来表示连续

几天的天气，对于噪波有很多算法，我使用的是最基础的柏林噪声来产生一系列1D的0-1的离散数值，然后使用线性插值的方法，获取这些离散点中任意位置的数值。用这样的方法可以生成无穷多个连续且随机的数值，既满足了天气的随机变化，也满足了均匀变化的要求。

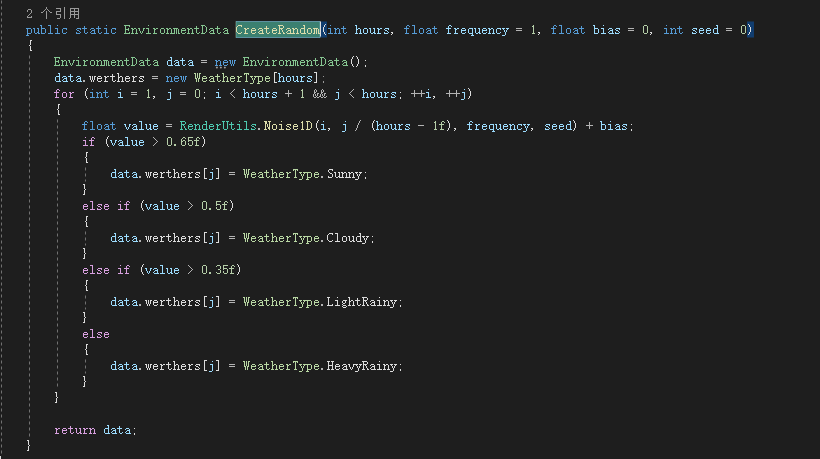


图 6-14 天气生成函数



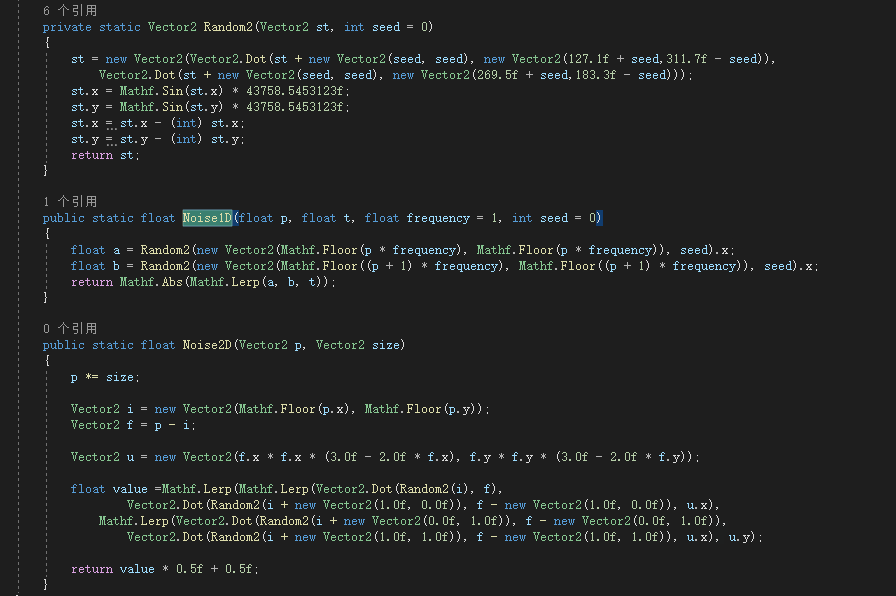


图 6-15 噪波生成函数

* 1. 点选功能

点选是玩家点击鼠标或者点击手机屏幕，可以选择到屏幕中指定物体的功能，当玩家想要移动到指定位置或者和指定NPC交互时，需要使用该功能来实现。

点选功能的底层逻辑是从摄像机向场景中发射射线，检测射线与场景中物体AABB的相交情况。游戏中使用的摄像机一般为透视摄像机，透视摄像机可以实现模拟人眼看到的近大远小的情况，透视摄像机使用平截头体来表示可视范围。平截头体如下图所示：



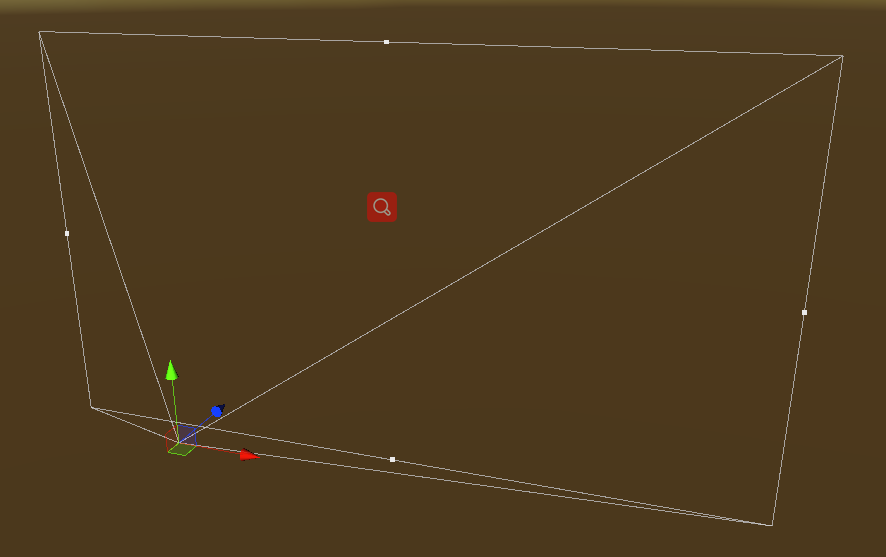


图 6-16 平截头体线框图

平截头体主要有近平面和远平面以及两个平面中间的其余四个平面组成，摄像机发出射线就是从近平面向远平面发射的射线，该射线接触到的第一个AABB就是点选目标的AABB，获取目标AABB同样需要通过场景管理器获取叶子节点的AABB，射线与AABB相交算法这里使用Slabs method算法，射线上的点满足P = o + t \* d,同时该点满足AAB B六个面的平面方程，ax + by + cz + d = 0,AABB的轴向与坐标系平行，所以简化为x

=d，y=d，z=d，联立方程求得t。

* 1. 界面功能

游戏内存在很多界面，包括背包界面，对话界面，交易界面等等，不同的界面需要实现不同的功能。背包界面与背包功能挂钩，需要读取道具表的数据；对话界面需要读取对话剧情表的数据；交易界面也需要读取道具表。

背包界面打开时获取当前背包内道具数量以及道具信息，根据这些信息给每一个道具创建一个单独的道具界面，每个道具界面显示该道具的缩略图，当右键点击该道具时可以出现对话框，选择不同的功能。道具界面的截图如下所示：



图 6-17 背包界面

对话界面通过玩家输入点选操作，如果玩家点选的目标是一个NPC，那么会打开跟NPC对话的界面，从对话表里查找当前对话目标的ID，根据目标ID查找当前对话的内容，对话界面如下所示：

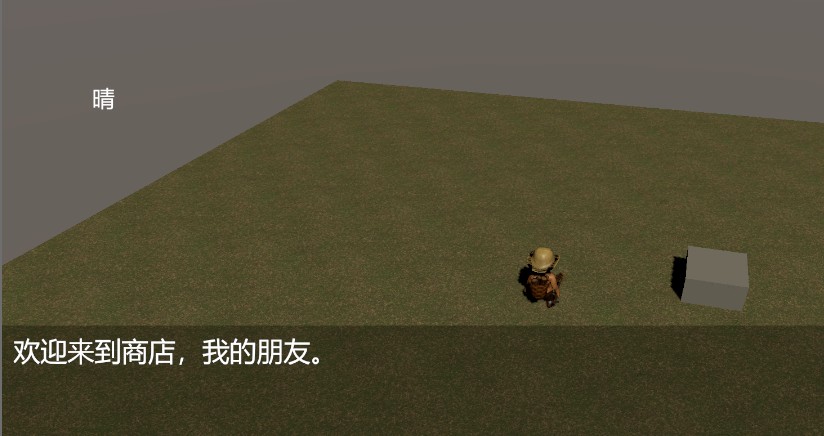


图 6-18 对话界面

交易界面出现在对话界面关闭之后，当玩家选择的目标是一个商店类型NPC时， 会打开商店界面，商店本身就是一个背包界面，然后也会打开角色的背包界面，购买物品的逻辑就是向玩家背包中添加一个目标物品的实例。交易界面如下：



图 6-19 交易界面

* 1. 自定义农场玩法

之前介绍了自定义农场的功能，通过类图和时序图简要说明了自定义农场的逻辑流程，这里详细介绍一下该玩法的具体逻辑。

在场景编辑完成之后，场景里会有一块长x，宽y的地面，继续分割地面成多个格子，假设每个格子的长宽均为0.5，那么一块地面上就会有2x\*2y个格子，将这些格子按行的顺序进行排列。存储第一个格子的坐标(x0, y0)，那么第(i,j)个格子的坐标为(x0 + 0.5 \* i, y0 + 0.5 \* j)，反之根据当前坐标也可以推算出当前坐标所处的格子。

为了方便玩家了解可编辑地块的范围，这里使用RenderManager在可以编辑的范围上绘制出一个矩形线框图来提示玩家，绿色的小矩形表示玩家当前选定的格子，打开背包界面后，选择铲子道具便可以对当前地块进行开垦，开垦完的格子可以种植种子。

自定义农场截图如下：



图 6-20 自定义农场截图

* 1. 农作物生长玩法

种植在地里的农作物接受游戏逻辑管理，会每帧更新自身状态，实现自身生长的属性以及外观变化。

农作物生长的相关属性包括生命值、生命上限、灌溉值、灌溉上限、饱食度、饱食上限、种植时间、成熟时间，这些数值相互作用。当农作物种在地里时，会记录种植农作物时的时间，每帧会计算生长时间t = 当前时间-种植时间，当t=成熟时间时，农作物生长完成，根据当时农作物的生命值分为三种情况健康的、不健康的、死亡三种状态，当前生命值大于生命上限的50%时是健康状态，1%-50%之间是不健康状态，0%是死亡状态，如果是健康状态和不健康状态可以进行收获，如果是死亡状态可以直接铲除。灌溉值反映农作物是否需要灌溉，当灌溉值小于渴水上限的一半时，则需要进行灌溉，不然会持续降低农作物生命值，饱食度同理。

两个数值同时影响农作物的生命值，通过生命值控制种子能否正常生长，需要玩家对农作物进行灌溉和施肥保证农作物健康成熟。

农作物生长截图如下：



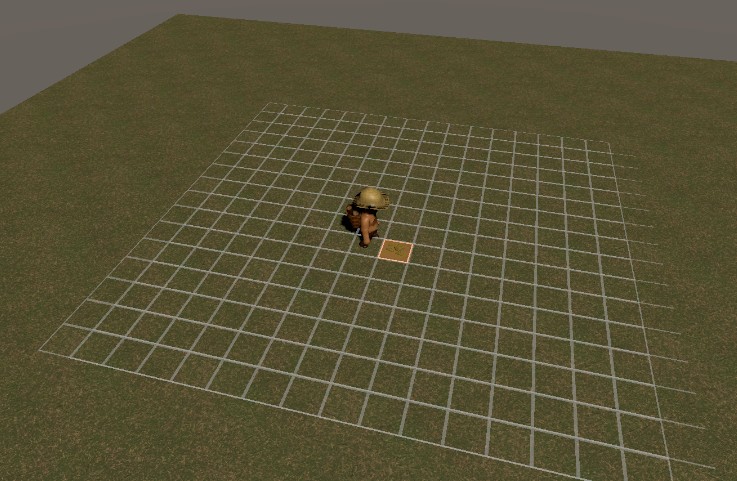


图 6-21 种子状态截图

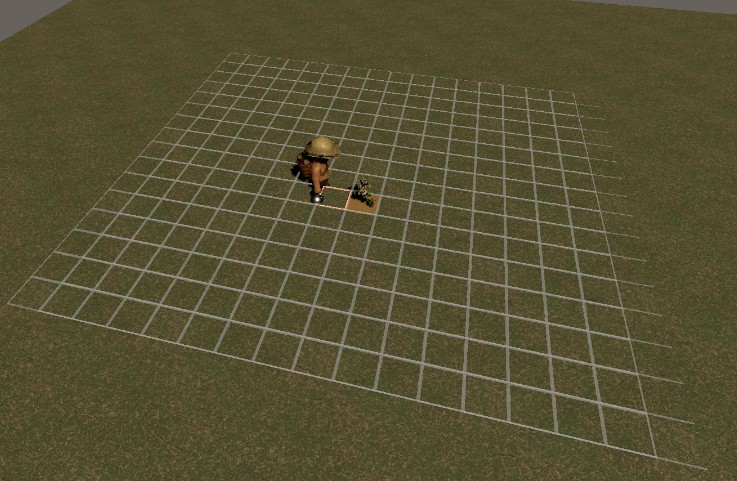


图 6-22 成熟状态截图

当农作物成熟之后，使用镰刀道具可以对农作物进行收割，收割完毕会在背包中 添加一个农作物实例。玩家可以对其进行贩卖，积累金钱，购买不同的种子进行种植， 最后打造属于自己的农场。

截图如下所示：



图 6-23 收获进背包

# 总结与展望

* 1. 本文总结

本系统使用Unity游戏引擎，编写了大量的底层功能，包括各种编辑器，管理器， 逻辑系统，数据库系统等等，在此之上实现了数个游戏的逻辑功能。使用多种技巧来提高游戏的运行效率，对于游戏来说，效率就是生命，我们需要尽一切可能去提高游戏的运行效率。Unity引擎本身封装了跨平台发布，跨平台Shader编译，并提供了良好的编辑器界面，对于提升游戏制作效率是非常有帮助的。

* 1. 不足之处

由于时间原因，资源制作的非常简陋，游戏玩法本身还有很多可以去提炼、深化的地方，玩法过于简单，目前没有时间去做更多的完善。但是只有发现问题面对问题才有可能解决问题，后续将对该系统进一步的改良。

* 1. 系统的展望

对于该项目有很多有意思的玩法设计，比如动物伙伴，好友，拍卖地块，购买房屋，摆设等等。后续有时间会进行细化和逐步完善，一步一步将他打造成一个沉浸感十足的有诚意的游戏。

# 致 谢

历时将近三个月的时间，本次毕业论文终于圆满结束，毕设期间由于老师和同学的督促指导，我完成了毕业项目的设计，论文编写和论文整理。期间遇到很多困难和障碍，在同学和老师的帮助下都一一度过。

感谢我的论文指导老师李耀老师，在此过程中得到了李耀老师的大力支持，尤其是在论文编写初期和论文修改期间，给了我很多关于论文的建议和意见，在他的耐心指导下，使我更加积极的完成毕业设计，并在最后完成了我的论文。通过这次毕业设计，让我对毕设过程中的学习内容有了更深的理解，同时也学到了很多课本上学不到的东西，积累了很多的技术经验，真正想实现一个软件要考虑方方面面。在这里再次对李耀老师对我的悉心的指导致以诚挚的谢意

# 参考文献

[1] 李宁.Android/OPhone 开发完全讲义[M].中国水利水电出版社.2010. [2] 胡伟.Android 系统架构及其驱动研究[J].广州广播电视大学学报.2010.

1. 郭志宏.Android 应用开发详解[M].电子工业出版社.2010.[2]杨丰盛.Android 应用开发揭秘[M].机械工业出版社.2010.
2. 高焕堂.Android 应用框架原理与程式设计 36 技[M/OL]. Google 公司,2008:132-150.
3. 马越.Android 的架构与应用[D].北京：中国地质大学(北京)硕士学位论文，

2008:9-20.

1. 《Beginning Android》 Mark L. Murphy 著 [7] 开放手机联盟．Android[DB/OL]．2008-07-30．
2. 李刚．Struts2 权威指南——基于 WebWork 核心的 MVC 开发[M].北京：电子工业出版社，2007．
3. 《Unlocking Android》 W.FRANK ABLESON，CHARLIE COLLINS，ROBI SEN 著[10] 《android 开发入门指南》（第 2 版） 索盖林 著
4. 陈天河.Struts,Hibernate,Spring 集成开发宝典[M]．北京：电子工业出版社，

2007．

1. 邓子云．基于 Struts2+Hibernate3+Spring2 的物流数据交换平台[J]．计算机应用与软件，2009，26(10)：88-100．
2. 曹大有．基于 Struts 和 Spring 框架技术的 Web 应用的开发过程[J]．计算机应用与软件，2008，25(8)：203-204．
3. 赵少卡．基于 Struts+Spring+Hibernate 架构的信息交流平台的设计与实现

[J]．计算机与现代化，2009(2)：62-66．

1. 刘晓华等著.J2EE 企业级开发.北京：电子工业出版，2003．23-36.