同济大学

实验报告

项目说明文档

程序设计范式

——星露谷风格的农场生活模拟游戏

小 组 成 员：王小萌、刘彦含、陈美希、邓语乐

指 导 教 师：赵钦佩

学院、 专业：计算机科学与技术学院 软件工程

完 成 日 期：2024.12.23

同济大学

Tongji University

目录

1. 项目简介

本项目是一款农场生活模拟游戏，旨在为玩家提供一个全方位的农场经营体验。玩家将负责管理自己的农场，从种植作物、养殖动物到与小镇居民进行交互。项目核心功能包括农场管理、社区交互、探索冒险以及角色成长和技能发展。玩家需要精心耕种、合理分配资源，同时与镇民建立关系，参与社区活动，接受任务以提升声望。此外，游戏还鼓励玩家探索周边环境，挖掘资源，提升生存技能。同时我们也初步实现了市场经济系统，以玩家的资源拥有量为评判指标对该种资源的市场价格进行调整，增添更多互动性和现实感。

2. 成员分工

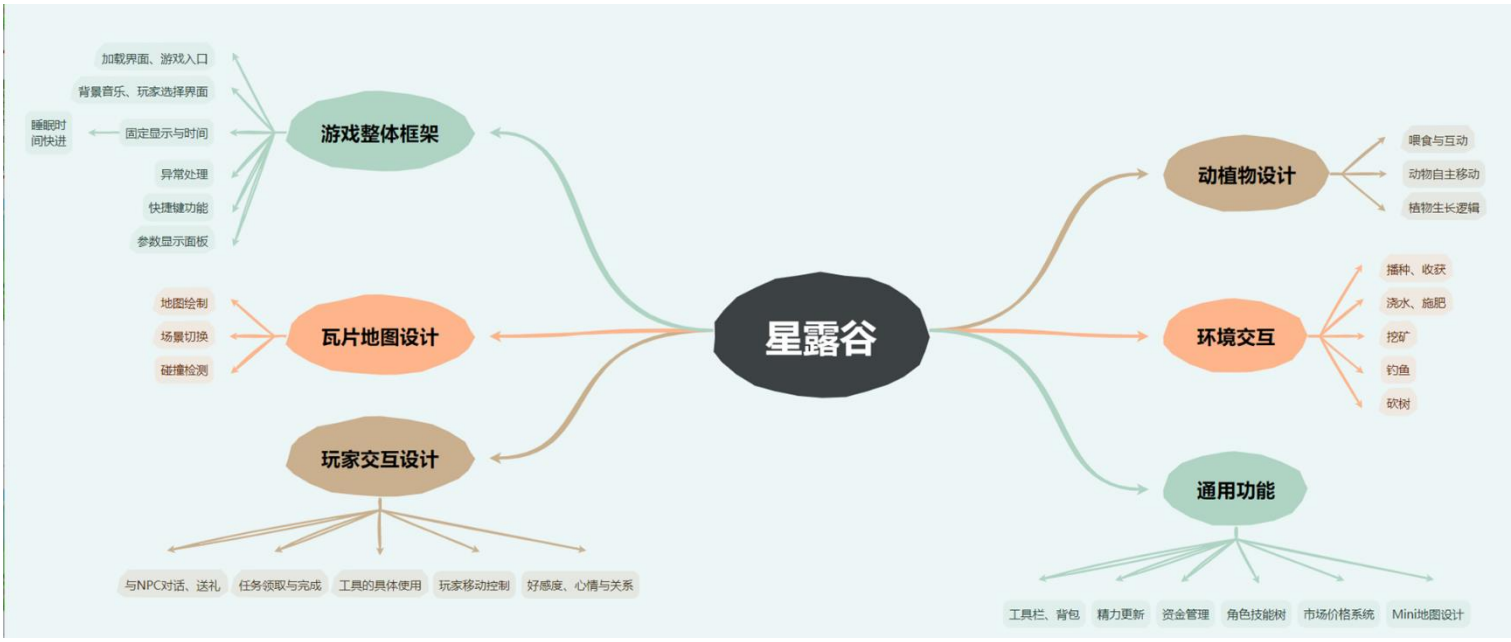
学号	姓名	分工	占比
2351882	王小萌	游戏整体架构实现、角色选择、背包工具栏实现、工具交互设计修改、资源管理、资金管理、UI管理、NPC交互修改、Mini地图显示、种子种植、异常处理	30%
2351591	刘彦含	动物养殖、农作物种植功能实现、汇报PPT制作、项目说明文档撰写、Git仓库管理、资金管理面板、工具交互、角色技能成长、市场价格系统、睡眠功能	30%
2354178	陈美希	地图设计与制作、碰撞检测、部分场景切换、NPC交互修改、NPC与任务系统关联	20%

2351273	邓语乐	工具交互设计、玩家整体系统设计、接受任务设计、NPC交互设计、工具操作及角色动作设计	20%

3. 项目C++特性的使用

在本次游戏设计的过程中，我们主要使用了STL容器、类和多态、模板、异常等C++特性。其中，我们主要使用了vector容器来实现对象容量的扩展；使用了类来封装各种函数功能的实现，保证了项目的安全性；同时也添加了多处异常处理来保证程序的正常稳定运行。

4. 项目实现功能概述



4.1. 基础功能

首先，我们实现了基本的农场管理，由玩家负责农场的日常运作，包括耕种、种植和收获作物；负责养殖牛、鸡、羊、猪、狗、猫等动物。此外，玩家可以进行浇水、施肥等操作来防止作物死亡。同时，玩家也可以进行资源管理，进行多种资源和工具的存储。其次，我们也提供了与镇上居民建立友谊和浪漫关系的社交功能。玩家可以与居民进行对话，接受并完成居民的委托任务，提高玩家的声望，获得奖励。另外，玩家有机会探索农场周边的自然环境，包括森林、池塘和神秘洞穴。在探索过程中，玩家可以挖掘矿物、钓鱼和砍树等，为游戏增添了冒险元素。而对于玩家个体而言，游戏设有角色技能树，涵盖农业、采矿、钓鱼和养殖业等多个领域；玩家可以通过进行相应的活动来提升对应经验值。

4.2. 扩展功能

除基础游戏功能外，我们还实现了一些扩展功能以增强玩家的游戏体验。其中，玩家可以随时在床上睡觉，并自行决定何时起床，起床后时间更新到第二天。而市场经济系统通过模拟现实世界的市场动态，允许产品的销售价格根据供需关系实时变动，增加了游戏的策略性，也使得农场经营更加真实和具有挑战性。此外，我们依托瓦片地图提供了丰富的环境交互功能，确保游戏世界的连贯性和探索的流畅性，在碰撞检测的基础上丰富了玩家的农场生活，也提供了资源收集和技能提升的机会，增加了游戏的深度和可玩性。另外，我们实现了包括工具栏、背包管理、精力更新、好感度系统、心情与关系管理、资金管理以及角色技能树等一系列通用功能，为玩家提供角色成长和资源管理机制，使游戏不仅是简单的农场经营，而且包含了角色发展和社交互动的多维度体验。

5. 项目功能技术细节

5.1. 基础功能

5.1.1. 农场经营

```
void Crop::update(float dt) {
    timeElapsed += dt;
    if (MapManager::getInstance()->currentMapLabel == 1 && this)
        this->setVisible(true);
    else
        this->setVisible(false);
    if (timeElapsed >= 1.0f) {
        age += timeElapsed;  // 增加经过的时间（以秒为单位）
        lastWateredTime += timeElapsed;  // 增加自上次浇水以来的时间
        timeElapsed = 0.0f;
        if (lastWateredTime >= 30)
            times++;
    }
    if (age >= 10) {  // 50秒等于生长周期
        grow();  // 调用生长函数
        age = 0;  // 重置年龄计时器
    }
    if (lastWateredTime >= 30) {  // 150秒未浇水
        state = CropState::Dead;
        changeTexture("../Resources/dead.png");  // 显示死亡阶段的图片
        deadSprite = Sprite::create("../Resources/dead.png");
        if (!deadSprite)
            throw("Failed to create dead sprite!");
        deadSprite->setPosition(Vec2(10, 10));  // 设置"dead"图片的位置与作物相同
        this->addChild(deadSprite);  // 将"dead"图片添加为Crop的子节点
        if (times > 3) {
            if (deadSprite) {
                deadSprite->removeFromParentAndCleanup(true);
                deadSprite = nullptr;  // 防止后续代码再次访问已删除的对象
            }
            if (gameScene)
                gameScene->removeCrop();  // 调用 GameScene 的 removeCrop 方法
        }
    }
}
```

```
void Crop::grow() {
    if (state != CropState::Mature && state != CropState::Dead) {  // 如果作物尚未成熟或死亡
        CCLOG("Into grow!");
        if (watered) {
            newAge += age;
            // 重置浇水状态
            watered = false;
            // 计算生长阶段，每10秒增加一个阶段
            int stage = (newAge / 10) % stageTextures.size();
            CCLOG("Stage: %d", stage);
            // 更新作物图片
            changeTexture(stageTextures[stage]);
            // 更新最后浇水的时间
            lastWateredTime = 0; // 重置浇水计时器
            if (stage == 6)
                state = CropState::Mature;
            else
                state = CropState::Growing;
        }
    }
    // 如果作物成熟，不再生长
    if (state == CropState::Mature) {
        std::string imagePath = "../Resources/mature_.png";
        Sprite* matureSprite = Sprite::create(imagePath);
        if (matureSprite) {
            // 设置图片的位置，这里以动物精灵的中心为位置参考
            matureSprite->setPosition(myCrop->getPosition());
            // 将图片添加到动物所在的节点
            this->addChild(matureSprite, 1, "matureSprite"); // 设置标签"happySprite"
            // 设置一个延时来移除图片，防止它一直显示
            this->scheduleOnce(CC_SCHEDULE_SELECTOR(Crop::removeSprite), 2.0f); // 2秒后移除图片
        }
    }
}
```

在进行作物种植时，我们根据游戏的时间流逝速度对时间进行累积，直到达到植物的生长周期并且玩家进行了浇水操作后农作物才可以正常生长；当玩家忘记浇水时，如果达到了农作物对应的三个生长周期均未浇水，则农作物因干旱死亡，弹出对应图标显示。在农作物正常生长时，根据预先设定的图片

路径进行纹理切换，实现农作物生长的可视化与动态显示；当作物成熟时，通过弹出图标提醒玩家进行收获，若玩家不及时进行收获，成熟后的农作物也会因为干旱而死亡，此时玩家则无法获得对应产物。

在进行动物养殖时，我们初始给予玩家六只不同类型的动物以供进行养殖体验。通过对动物移动方向的判断进行相应的纹理切换，我们实现了动物不同方向前进时的转身效果。另外，我们采用了与玩家移动控制不同的逻辑进行动物移动，按照预先设定好的路径在指定区域内进行移动操作，实现了动物的动态展示。另外，对于玩家与动物的交互，我们设计了喂食与抚摸互动，分别可以增加对应动物的健康值、心情值和好感度。玩家可以通过在动物身边右击鼠标展示动物的各种数据的显示面板，直观显示所拥有动物的不同属性。当动物的健康值和心情值满时，在玩家与动物进行互动的过程中，玩家可以获得每种动物不同的产物，不同的产物具有不同的售价，可在商店中进行售卖。

```
void Animal::moveAlongPath(const std::vector<Vec2>& path, const std::string& picturename) {
    // 使用 ActionInterval 作为 action 变量的类型
    ActionInterval* action = nullptr;
    for (size_t i = 0; i < path.size() - 1; ++i) {
        // MoveTo::create 返回的是 FiniteTimeAction 类型，需要转换为 ActionInterval 类型
        auto moveAction = static_cast<ActionInterval*>(MoveTo::create(2.0f, path[i + 1]));
        // 使用 Sequence::createWithTwoActions 创建 Sequence，需要两个 FiniteTimeAction 参数
        action = (i == 0) ? moveAction : Sequence::createWithTwoActions
        (static_cast<FiniteTimeAction*>(action), static_cast<FiniteTimeAction*>(moveAction));
    }
    // RepeatForever::create 需要一个 ActionInterval 参数
    auto repeatAction = RepeatForever::create(action);
    an_sprite->runAction(repeatAction);
    // 需要确保运动过程中不停地更新方向（即精灵帧）
    schedule([this, picturename](float dt) {
        // 在运动时更新方向，并传递图片名称
        updateDirection(dt, picturename);
        }, 1.0f / 60.0f, "update_direction_key");
}

void Animal::openAnimalMenu() {
    // 获取当前屏幕的可视区域大小
    Size visibleSize = Director::getInstance()->getVisibleSize();
    // 创建一个层作为菜单的容器
    menuLayer = Layer::create();
    menuLayer->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    // 创建背景图片精灵，并设置锚点为(0.5, 0.5)以居中显示
    auto background = Sprite::create("an_menu.png");
    background->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    background->setAnchorPoint(Vec2(0.5, 0.5));
    background->setScale(2.5); // 设置图像大小为原来的1.5倍
    menuLayer->addChild(background);
    // 创建一个标签，显示标题
    auto label = Label::createWithTTF("Animal Information", "fonts/Marker Felt.ttf", 30);
    label->setColor(Color3B::BLACK);
    label->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4 + 196)); // 调整标签位置
    menuLayer->addChild(label);
    // 创建一个按钮，点击后关闭菜单
    auto closeBtn = MenuItemImage::create(
        "close.png", // 正常状态的图片
        "close.png", // 选中状态的图片
        CC_CALLBACK_1(Animal::closeAnimalMenu, this));
    closeBtn->setScale(2.5); // 设置图像大小为原来的1.5倍
    closeBtn->setPosition(Vec2(visibleSize.width / 4 + 65, visibleSize.height / 4 + 32)); // 调整关
    // 创建一个菜单对象，并添加关闭按钮
    auto menu = Menu::create(closeBtn, NULL);
    menu->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    menuLayer->addChild(menu);
    // 添加背景边框的显示
```

5.1.2. 社区交互

在与小镇居民进行交互的过程中，我们实现了与小镇居民进行对话、领取与完成任务、赠送礼物等

功能。当玩家接近居民时，与居民距离较近时鼠标移动到居民身上会显示不透明的对话气泡，当远离时会变为半透明状态。当靠近并左击小镇居民时可触发与其的对话内容，在对话过程中实现任务领取；当任务完成时再次对话可以收获居民的夸赞。当任务领取并完成后再次与居民对话，则会随机进行日常对话与聊天，实现了基本的社区交互功能。

```
for (auto npc : npcs) {
    if (!npc) continue;
    // 获取 NPC 的包围框
    Rect npcWorldBoundingBox = npc->getBoundingBox();
    // 判断鼠标是否在 NPC 包围框内
    if (npcWorldBoundingBox.containsPoint(mousePosInDesign)) {
        // 鼠标在NPC上
        updateDialogPosition(npc);
        float distance = getPlayerPosition().distance(npc->getPosition());
        if (distance < 50.0f)
            // 离得近：不透明
            uiManager->dialogSprite->setOpacity(NO_TRANSPARENT);
        else
            // 离得远：半透明
            uiManager->dialogSprite->setOpacity(HALF_TRANSPARENT);
        foundNpcUnderMouse = true;
        break;
    }
}
 for (auto npc : npcs)
if (!foundNpcUnderMouse) {
    uiManager->dialogSprite->setOpacity(0); // 隐藏对话框
}
```

```
// 计算对话框的位置
void Player::updateDialogPosition(NPC* npc) {
    // 获取 NPC 的位置（世界坐标）
    Vec2 npcWorldPosition = npc->getParent()->convertToWorldSpace(npc->getPosition());
    // 获取 NPC 的大小
    Size npcSize = npc->getContentSize();
    // 计算对话框的位置（相对于世界坐标）
    Vec2 dialogPosition = npcWorldPosition;
    // 如果对话框是相对于场景或地图的，直接设置对话框的位置
    uiManager->dialogSprite->setPosition(dialogPosition);
}
```

5.1.3. 冒险探索

```
void Tool::usetool() {
    auto player = Player::getInstance(selectedCharacter, nickname);
    auto playerPos = Player::getInstance(selectedCharacter, nickname)->getPosition();
    auto direction = Player::getInstance(selectedCharacter, nickname)->_currentDirection;
    Vec2 dstPos;
    switch (direction) {
    case 0://下
        dstPos = Vec2(playerPos.x, playerPos.y - 16);
        break;
    case 1://右
        dstPos = Vec2(playerPos.x + 16, playerPos.y);
        break;
    case 2://上
        dstPos = Vec2(playerPos.x, playerPos.y - 16);
        break;
    case 3://左
        dstPos = Vec2(playerPos.x - 16, playerPos.y);
        break;
    }
    switch (type) {
    case ToolType::HOEPLUS:
    case ToolType::HOE: // 锄头功能：挖坑
        if (MapManager::getInstance()->getCurrentBlockLabel() == 7) { //挖矿
            if (playerPos.x < 600 && playerPos.x>460 && playerPos.y < 400 && playerPos.y>300)
                ItemManager::getInstance(selectedCharacter, nickname)->addItem(Item::ItemType:
                player->changeMining();
            }
        }
        break;
    case ToolType::AXEPLUS:
    case ToolType::AXE: // 斧头功能：砍树
        if (MapManager::getInstance()->getCurrentBlockLabel() == 2){ ... }
        break;
    case ToolType::WATERING_CANPLUS:
    case ToolType::WATERING_CAN: // 水壶功能：浇水
        player->changePlanting();
        break;
    case ToolType::FISHING_RODPLUS: // 鱼竿功能：钓鱼
    case ToolType::FISHING_ROD: // 鱼竿功能：钓鱼
        if (MapManager::getInstance()->getCurrentBlockLabel() == 4){ ... }
        break;
    case ToolType::FERTILIZER: //
        break;
    case ToolType::ANIMALFOOD: //
        player->changeBreeding();
        break;
    default:
        break;
    }
}
```

在进行冒险探索时，我们根据所选择的不同工具以及当前所处的地图界面分别进行不同的探索操作。当手持镐头且与矿石距离较近时玩家通过鼠标左键使用工具，即可获得矿石，背包中的矿石总数加一；当手持鱼竿且靠近鱼塘时，玩家同理使用工具获得鱼；同理手持斧头靠近树木时，玩家使用工具进

行砍树，可以实现挥动斧头的动效，获得木头，背包中的木头总数加一。在不同的地图板块中通过使用不同的工具，玩家可以实现对于不同资源的获取，与此同时可以提升玩家的相关经验值，并在快捷键控制的菜单中进行可视化，便于玩家查看自己的当前属性并进行针对性地提升，实现了游戏设计的个性化，具有多种游戏进行模式的可能。

5.1.4. 角色个人成长

```
void Player::openPlayerMenu() {
    Size visibleSize = Director::getInstance()->getVisibleSize();
    menuLayer = Layer::create();
    menuLayer->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    auto background = Sprite::create("an_menu.png");
    background->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    background->setAnchorPoint(Vec2(0.5, 0.5));
    background->setScale(2.5); // 设置图像大小为原来的1.5倍
    menuLayer->addChild(background);
    auto label = Label::createWithTTF("Player Information", "fonts/Marker Felt.ttf", 30);
    label->setColor(Color3B::BLACK);
    label->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4 + 196)); // 调整标签位置
    menuLayer->addChild(label);
    auto closeBtn = MenuItemImage::create(
        "close.png", // 正常状态的图片
        "close.png", // 选中状态的图片
        CC_CALLBACK_1(Player::closePlayerMenu, this));
    closeBtn->setScale(2.5); // 设置图像大小为原来的1.5倍
    closeBtn->setPosition(Vec2(visibleSize.width / 4 + 65, visibleSize.height / 4 + 32)); // 调整关闭按钮位置
    auto menu = Menu::create(closeBtn, NULL);
    menu->setPosition(Vec2(visibleSize.width / 4, visibleSize.height / 4));
    menuLayer->addChild(menu);
    Sprite* playerImage;
    if (_selectedCharacter == 1) {
        playerImage = Sprite::create("player1.png");
    }
    else {
        playerImage = Sprite::create("player2.png");
    }
```

```
playerImage->setAnchorPoint(Vec2(0.5, 0.5)); // 设置锚点为图像中心
playerImage->setScale(4.5); // 设置图像大小为原来的1.5倍
menuLayer->addChild(playerImage);
auto labelPlanting = Label::createWithTTF("Planting_Skills: " + std::to_string(this->getPlanting()) + " / 100", "fonts/Marker Felt.ttf", 50);
if(this->getPlanting()!=100)
    labelPlanting->setColor(Color3B::WHITE);
else
    labelPlanting->setColor(Color3B::RED);
labelPlanting->setPosition(Vec2(590-200, 445- 175));
menuLayer->addChild(labelPlanting);
auto labelBreeding = Label::createWithTTF("Breeding_Skills: " + std::to_string(this->getBreeding()) + " / 100", "fonts/Marker Felt.ttf", 50);
if (this->getBreeding() != 100)
    labelBreeding->setColor(Color3B::WHITE);
else
    labelBreeding->setColor(Color3B::RED);
labelBreeding->setPosition(Vec2(590 - 200, 385 - 175));
menuLayer->addChild(labelBreeding);

auto labelMining = Label::createWithTTF("Mining_Skills: " + std::to_string(this->getMining()) + " / 100", "fonts/Marker Felt.ttf", 50);
if (this->getMining() != 100)
    labelMining->setColor(Color3B::WHITE);
else
    labelMining->setColor(Color3B::RED);
labelMining->setPosition(Vec2(590 - 200, 320 - 175));
menuLayer->addChild(labelMining);
auto labelFishing = Label::createWithTTF("Fishing_Skills: " + std::to_string(this->getFishing()) + " / 100", "fonts/Marker Felt.ttf", 50);
if (this->getFishing() != 100)
    labelFishing->setColor(Color3B::WHITE);
else
    labelFishing->setColor(Color3B::RED);
labelFishing->setPosition(Vec2(590 - 200, 260 - 175));
menuLayer->addChild(labelFishing);
// 将菜单层添加到场景
auto scene = Director::getInstance()->getRunningScene();
scene->addChild(menuLayer, 10); // 设置一个合适的层级，确保菜单层在最上面
```

玩家角色初始时具有多种属性，如钓鱼技能、种植技能、养殖技能、挖矿技能等。为了便于直观评估角色当前的多种状态，我们设计了一个与动物属性显示类似的玩家属性介绍界面，为了防止玩家角色工具使用等多种鼠标操作与之发生冲突，我们选择了使用快捷键P打开角色属性面板。当玩家初始进入游戏时，各种技能值均较低，玩家可以通过从事相关的活动获取经验值，提升自己的技能。当玩家的某一技能达到100时，再次打开玩家属性界面，该属性对应的标识符变为红色，表示玩家已达到了该领域的优秀状态，技能较高超，从而实现了玩家角色的个人成长。

5.2. 扩展功能

5.2.1. Mini地图显示

```
void UIManager::toggleMiniMap(const Vec2& playerPos, const Size& mapSize) {
    isMiniMapVisible = !isMiniMapVisible;
    miniMap->setVisible(isMiniMapVisible);
    if (isMiniMapVisible) {
        auto screenSize = Director::getInstance()->getVisibleSize();
        float addWidth = 0, addHeight = 0;
        CCLOG("currentMapLabel:%d", mapManager->currentMapLabel);
        if (mapManager->currentMapLabel >= 1 && mapManager->currentMapLabel <= 4) {
            if (mapManager->currentMapLabel == 2 || mapManager->currentMapLabel == 4) {
                addWidth = screenSize.width;
            }
            if (mapManager->currentMapLabel == 1 || mapManager->currentMapLabel == 2) {
                addHeight = screenSize.height;
            }
            Vec2 normalizedPos((playerPos.x + addWidth) / (screenSize.width * 2), (playerPos.y + addHeight) / (screenSize.height * 2));
            playerMarker->setPosition(Vec2(miniMap->getContentSize().width * normalizedPos.x, miniMap->getContentSize().height * normalizedPos.y));
        }
        else if (mapManager->currentMapLabel == 5) {
            Vec2 normalizedPos(0.25, 0.8);
            playerMarker->setPosition(Vec2(miniMap->getContentSize().width * normalizedPos.x, miniMap->getContentSize().height * normalizedPos.y));
        }
        else if (mapManager->currentMapLabel == 6) {
            Vec2 normalizedPos(4.0 / 10.5f, 1.0 / 6.5f);
            playerMarker->setPosition(Vec2(miniMap->getContentSize().width * normalizedPos.x, miniMap->getContentSize().height * normalizedPos.y));
        }
        else if (mapManager->currentMapLabel == 7) {
            Vec2 normalizedPos(7.2 / 10.5f, 5.8 / 6.5f);
            playerMarker->setPosition(Vec2(miniMap->getContentSize().width * normalizedPos.x, miniMap->getContentSize().height * normalizedPos.y));
        }
    }
}
```

在玩家冒险探索的过程中，由于地图的分区与室内室外、矿山的多场景切换，为了确定玩家在世界地图的位置，我们设置了miniMap来显示玩家在世界地图上的位置，并用星星与醒目的红色label标记当前玩家位置，便于玩家进行全局把握自己所处位置并进行下一步目标地址的搜寻。其中，我们通过快捷键M切换显示状态。

5.2.2. 资金管理系统

```
int x = 0;
for (int i = 0; i < Items.size(); i++) {
    if (Items[i] && Items[i]->getBoundingBox().containsPoint(locationInItemsBg)) {
        if (event->getMouseButton() == EventMouse::MouseButton::BUTTON_LEFT) {
            CCLOG("Item[%d] selected and used", i);
            selectItem(i);  // 选中物品
            if (chest->isOpen == 1){ ... }
            else if (UIManager::getInstance(x, "")->isPriceBoardOpen == 1) {//买卖东西

                if (Items[i]->getType() != Item::ItemType::GIFT) {
                    Items[i]->decreaseQuantity(1);
                    UIManager::getInstance(x, "")->setMoney(Items[i]->price);
                }
            }
            else
                useitem();  // 直接使用物品
        }
        else if (event->getMouseButton() == EventMouse::MouseButton::BUTTON_RIGHT) {
            discardItem();  // 右键丢弃物品
        }
        return;
    }
}
```

初始时，我们对于每一个可供售卖的物品进行初始定价，并可以通过商店的价格表进行显示。当玩家拥有较多某种物品或者希望进行售卖获得金钱时可以到商店售出，对应物品数量减少，同时玩家拥有的金钱数量增加同等的金钱数；而当玩家的金钱足够多时，支持玩家到商店购买工具对当前设备进行升级，也支持购买礼物送给小镇上的居民，增加对应居民的好感度，当好感度达到某一设定值时可以与居民建立亲密关系。

5.2.3. 市场价格系统

```
void ItemManager::addItem(Item::ItemType type) {

    float gridWidth = 32.0f; // 物品栏宽度
    float startX = (Director::getInstance()->getVisibleSize().width - gridWidth * 10) / 2.0f;
    float startY = Director::getInstance()->getVisibleSize().height * 0.1f + 32.0f;

    for (int i = 0; i < Items.size(); i++) {
        if (Items[i] != nullptr && Items[i]->getType() == type) {
            // 如果物品栏中已经有这个类型的物品，增加数量
            Items[i]->increaseQuantity(1);
            if (type == Item::ItemType::MINERAL && Items[i]->getQuantity() >= 30) {
                Items[i]->price = 60;
            }
            else if (type == Item::ItemType::MINERAL && Items[i]->getQuantity() < 30) {
                Items[i]->price = 80;
            }

            return;
        }
```

```
void UIManager::showPriceBoard() {
    auto visibleSize = Director::getInstance()->getVisibleSize();
    itemManager = ItemManager::getInstance(selectedCharacter, nickname);
    // 创建价格表背景图片
    Sprite* priceListBg;
    if (itemManager->getItemQuantity(Item::ItemType::MINERAL) < 30) {
        priceListBg = Sprite::create("../Resources/priceBoard.jpg");
    }
    else {
        priceListBg = Sprite::create("../Resources/priceBoard_lower.jpg");
    }
    priceListBg->setPosition(visibleSize.width / 2, visibleSize.height / 2);

    priceListBg->setName("PriceBoard");
    this->addChild(priceListBg, 10);
    isPriceBoardOpen = 1;
    // 关闭按钮
    auto closeButton = MenuItemImage::create("../Resources/close.png", "../Resources/close.png",
        [=](Ref* sender) {
            this->closePriceBoard();
        });
    closeButton->setScale(4.0f);
    auto menu = Menu::create(closeButton, nullptr);
    menu->setPosition(priceListBg->getPositionX() + priceListBg->getContentSize().width / 4,
        priceListBg->getPositionY() + priceListBg->getContentSize().height / 4 );
    priceListBg->addChild(menu);
}
```

我们以矿石为例对市场价格体系进行初步模拟。主要根据矿石的产量（即拥有量）对矿石的售出价格进行动态判断。当玩家拥有的矿石数量小于30时，说明此时矿石较为珍贵，售出即可获得较高的收益，因此此时商店的价格表中矿石的价格较高；而当玩家拥有的矿石数量大于等于30时，说明此时矿石较为常见，其价值低于原始价值，玩家此时售出获得的收益较低。我们通过对于矿石产量的动态评估实现了对市场价格系统的初步模拟，而对于其他农作物与产物等，也可以采用类似的判断逻辑进行动态评估，展现市场价格的变化趋势。

5.2.4. 睡眠实现时间快进

```
// 初始化睡眠面板
sleepPanel = LayerColor::create(Color4B(20, 20, 20, 220)); // 深色且几乎不透明
sleepPanel->setVisible(false);
this->addChild(sleepPanel);
// 创建一个标签，显示标题
auto label = Label::createWithTTF("Sleeping...", "fonts/Marker Felt.ttf", 100);
label->setColor(Color3B::WHITE);
label->setPosition(Vec2(520,250)); // 调整标签位置
sleepPanel->addChild(label);
else if (mapManager->currentMapLabel == 5 && locationInWorld.distance(Vec2(521, 223)) < 50 && playerPos.distance(Vec2(521, 223)) < 50) {
    if (sleepPanel->isVisible()) {
        sleepPanel->setVisible(false);
        Director::getInstance()->resume();
        uiManager->currentEnergy = 100;
        uiManager->currentDay += 1;
        uiManager->currentHour = 8;
        uiManager->currentMinute = 0;
        uiManager->currentWeekday = (uiManager->currentWeekday + 1) % 7;

        if (uiManager->currentDay > uiManager->getDaysInMonth(uiManager->currentMonth)) {
            uiManager->currentDay = 1;
            uiManager->currentMonth += 1;
            if (uiManager->currentMonth > 12) {
                uiManager->currentMonth = 1;
            }
        }
    }
    else {
        sleepPanel->setVisible(true);
        Director::getInstance()->pause();
    }
}
```

当玩家的位置在家所处的地图块上且与床所在的坐标相接近时，玩家可以通过鼠标左键点击床的位置实现睡眠。当玩家处于睡眠状态时，整个游戏界面暂停，时间停止流逝，而在玩家再次点击床的位置时，玩家实现睡眠结束，此时开启新的一天，时间从第二天早晨八点开始继续流逝，精力值恢复满值，实现了时间的快进与玩家睡眠休息功能。

6. 项目使用说明

动物养殖：左键单击互动；拿着食物左键双击喂食；右击动物打开动物属性面板

植物种植：拿着水左击浇水；拿着饲料左击施肥

工具栏：左键使用；右键丢弃

商店：左键售卖，右键丢弃（除礼物），点击BUY按钮可以实现购买

快捷键：E暂停/继续；M开/关地图；C使用工具；V丢弃工具；P打开角色属性；F开/关任务窗口；Shift行走速度减半

玩家交互：左键优先出现任务对话，第二次点击询问是否完成任务，完成后给予奖励；再次与小镇居民对话则为随机日常对话，右键打开与小镇居民的好感度界面