

星露谷物语

黎江北 王云杰
李星钢 张亚东

C++特性使用

1. STL容器

```
void NPC::setPath(const  
std::vector<Vec2>& newPath)
```

2.c++11新特性

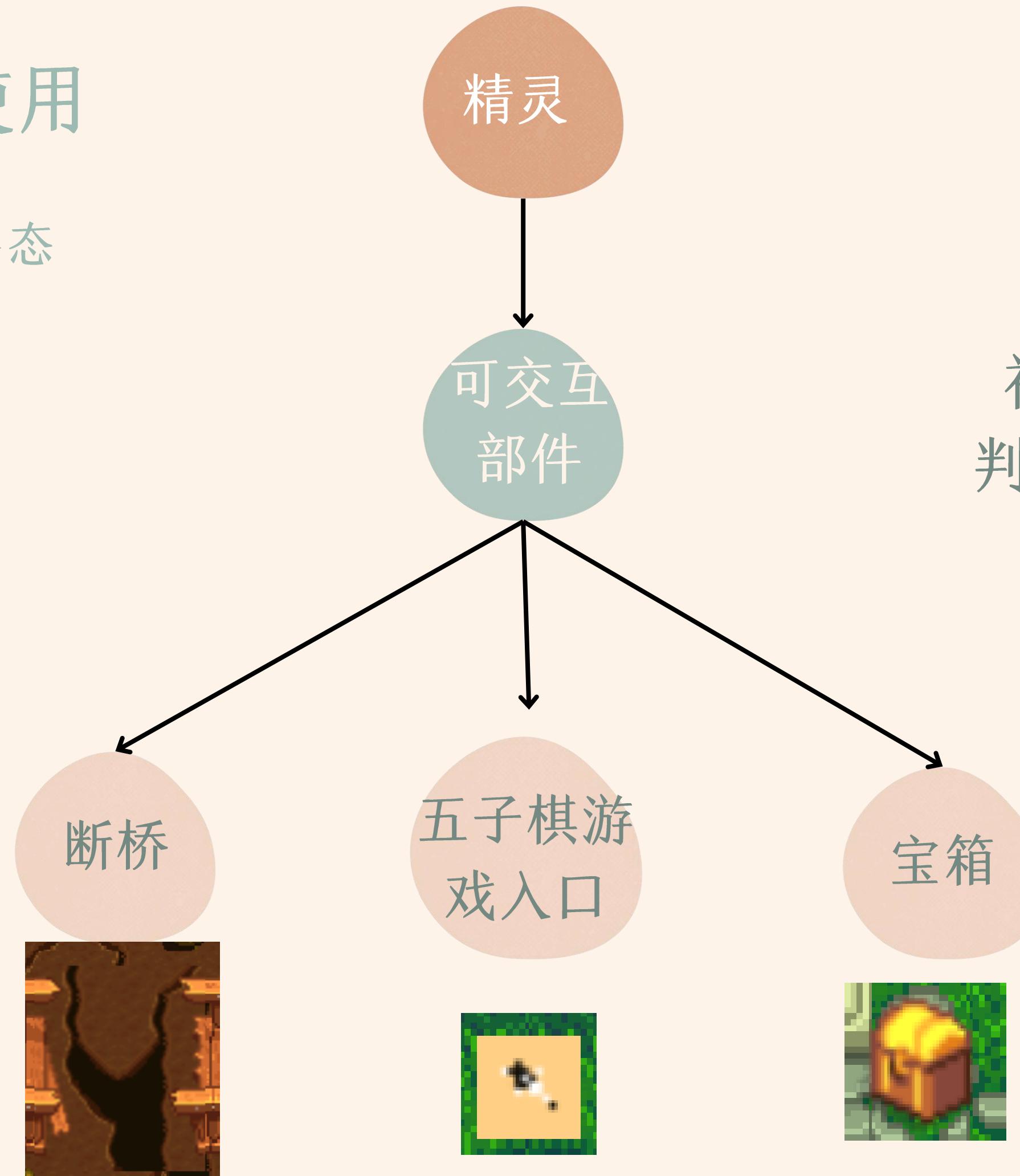
```
for (auto& i : NPCpath)  
    i *= 16;
```

3. 异常

```
try {  
    场景转换  
    if (超出现有地图  
        throw “前面的区域以后再来探索吧”  
    } catch (...) {  
        传回地图中央  
    }
```

C++特性使用

4.类和多态



初始化和
判断是否点
击

初始化时经单独处理后
调用父类的初始化

重写点击回调函数

PART 01

农场和牧场

1. 玩家可以耕种、种植并收获多种作物，作物的成熟时间根据种类和季节变化。
2. 养殖包括但不限于牛、鸡和羊等动物。
3. 农场操作包括浇水、施肥，以及管理干旱和病害威胁，未及时处理会导致作物死亡。
4. 管理农场资源，如水资源、种子和肥料存储。

01 牧场功能

模拟动物行走

在正常状况下，动物都会按照预设好的路线，在设定的区域内行走。

喂食系统

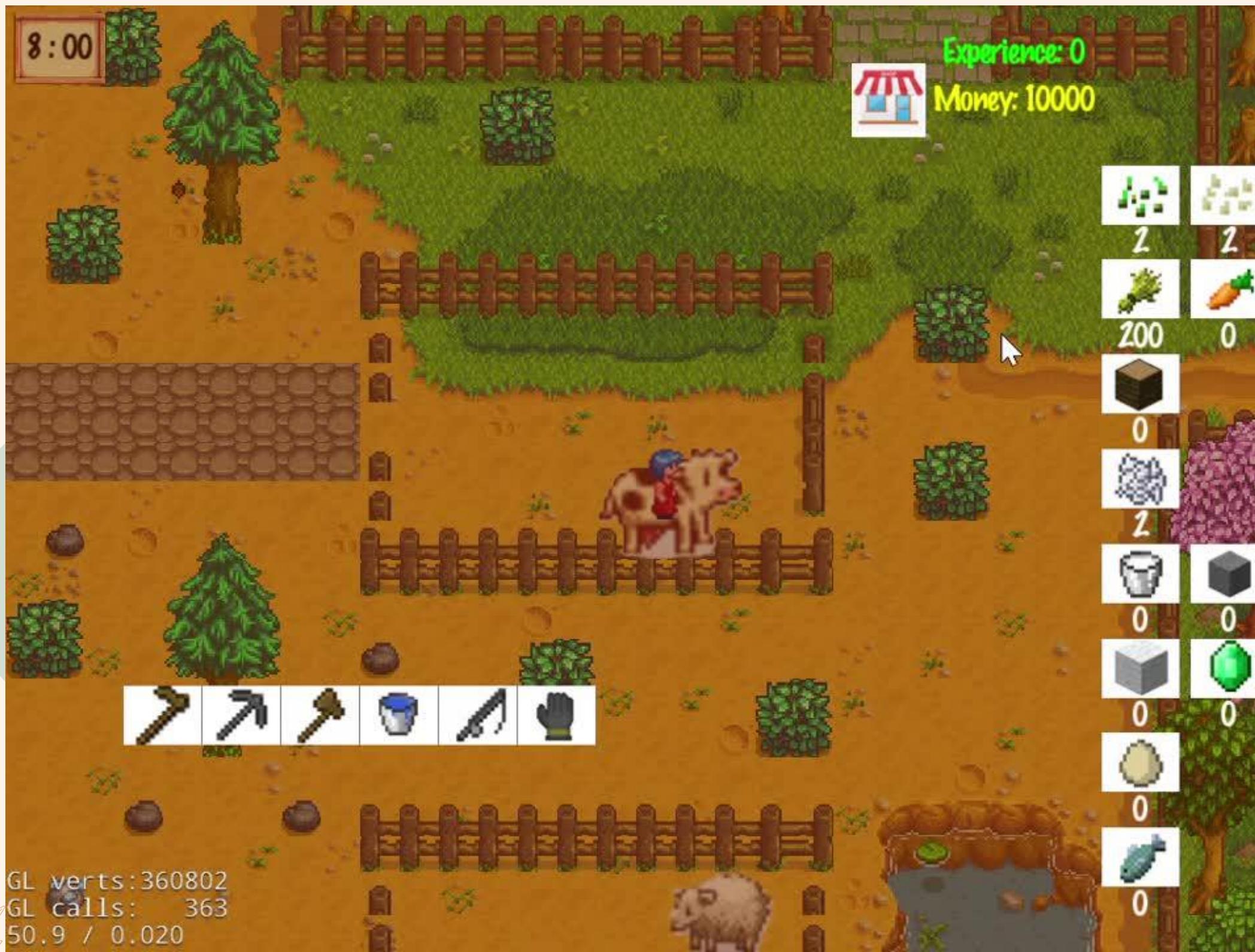
当主角靠近动物时按下E，就会弹出弹窗，喂养成功，消耗小麦一个并且获得对应的产物

幸福度系统

当主角靠近动物时按下Q，就会弹出弹窗，抚摸成功，动物的幸福度提示，当长时间不喂食和抚摸，则动物的幸福度会下降，此外幸福度还会受天气的影响，暴晒和雨天幸福度都会降低。

幸福度影响的是动物产出产物的数量，当幸福度最低时，只会产出一个作物，而后随着幸福度提升以此设定为两个，四个。

01 牧场功能



01 牧场代码实现

```
class Animal : public Sprite {
public:
    //Animal动画 动画路径示例: Animal/filename/moveup1.png
    Animate* moveup;
    Animate* moveleft;
    Animate* moveright;
    Animate* movedown;

    //标记Animal是否被选中
    bool ifSelected;

    //路径相关
    std::vector<Vec2> Animalpath;//路径参数
    int currentPathIndex;//路径索引
    float speed;//速度

    //Animal名称
    std::string animalName;

    //更新Animal的移动和动画
    void updatemove(float dt);

    // 播放动画
    void playAnimation(const std::string& direction);

    // 设置移动路径
    void setPath(const std::vector<Vec2>& path);
};
```

基类animal继承自sprite,包含四个方向的动画，bool型的判断符标记是否被选中，animalName指代动物的名字《我们看见的每只动物的移动路径由路径相关的三个参数来决定，uodate函数需要传入时间决定几秒更新一次，playAnimation从resources里选资源设定，setPath实现animal的具体行走功能。

01 牧场代码实现

```
void Animal::updatemove(float dt)
{
    if (Animalpath.empty()) return;

    // 获取目标位置
    Vec2 targetPosition = Animalpath[currentPathIndex];
    Vec2 currentPosition = this->getPosition();

    // 计算当前位置和目标位置的距离
    Vec2 direction = targetPosition - currentPosition;
    float distance = direction.length();

    // 如果到达目标，切换到下一个路径点
    if (distance < 1.0f) {
        currentPathIndex = (currentPathIndex + 1) % Animalpath.size(); // 循环路径
    }

    // 移动 Animal
    Vec2 moveDirection = direction.getNormalized();
    setPosition(currentPosition + moveDirection * speed * dt);

    // 根据方向播放动画
    if (fabs(moveDirection.x) > fabs(moveDirection.y)) { // 水平移动
        if (moveDirection.x > 0) {
            playAnimation("right");
        }
        else {
            playAnimation("left");
        }
    }
    else { // 垂直移动
        if (moveDirection.y > 0) {
            playAnimation("up");
        }
        else {
            playAnimation("down");
        }
    }
}
```

updatemove函数是该类的核心功能，在开始时先进行了判断，如果设置的路径为空，则直接返回，不继续进行，之后以此获取目的地位置和当前位置，计算两者的距离，在距离小于一时切换到下一条路径，之后开始移动animal，依照方向执行设定好的移动动画。

```
speed = 50.0f;//设置速度
currentPathIndex = 0;//初始路径
ifSelected = false;//标记Animal未被选中

// 设置位置更新函数
schedule([=](float dt) {updatemove(dt); }, 0.0f, "animal_updatemove_key");
```

调用时的实例如上，已一个调度器实现了该函数的定时重复执行，这样就实现了动物重复按照预设好的路径行走的功能。

01 牧场代码实现

```
class Cow : public Animal {  
public:  
    //Animal静止图片路径为 Animal/filename/static.png  
    static Cow* create(const std::string& filename);  
  
    //Animal精灵大小  
    static int Animalsize_x;  
    static int Animalsize_y;  
  
    //Animal纹理的动画顺序  
    static int Animalorder_up;  
    static int Animalorder_left;  
    static int Animalorder_right;  
    static int Animalorder_down;  
  
    //幸福度  
    double happiness = 0;  
  
    //主角  
    MainCharacter* mainChar;  
    //地图指针  
    TMXTiledMap* mainmap;  
  
    //cow已被喂养的反馈弹窗  
    cocos2d::Label* cow_feed_label;  
  
    //cow已被抚摸的反馈弹窗  
    cocos2d::Label* cow_touch_label;  
  
    //检测主角与动物的位置关系  
    void isMainCharNear(float delta);  
  
    //检测是否被抚摸  
    void isMainCharTouch(float delta);  
  
    //随时间幸福会降低  
    void decreaseHappiness(float delta);  
  
    //是否在附近  
    bool isNearSprite = false;  
  
    //添加键盘监听事件  
    void addKeyboardListener();  
  
    //当按键按下时产生的事件  
    void onKeyPressed(EventKeyboard::KeyCode keyCode, Event* event);  
  
    //标签隐藏  
    void hideLabel(float dt);  
  
    //初始化  
    bool init();  
  
    //设置主角  
    void setMaincharacter(MainCharacter* mainCharacter) {  
        mainChar = mainCharacter;  
    }  
  
    //传入地图指针  
    void setMap(TMXTiledMap* map) {  
        mainmap = map;  
    }  
  
    //播放动画  
    static void move(Cow* cow, TMXTiledMap* map);
```

三种动物的子类实现都大同小异，此处以Cow类为例，继承自Animal的父类，包含的辅助参数包括了具体的精灵大小和动画播放次序，主角的指针和地图的指针，以及bool型的isNearSprite判断主角与动物关系自己的属性则主要是幸福度，以及两种功能喂养和抚摸的弹窗。

函数方面包括了辅助功能的函数isMainCharNear判断

主角和动物的位置关系，addKeyboardListener和onKeyPressed实现按下键盘特点键启用某一功能，setMaincharacter和setMap传入指针，hideLabel用于隐藏喂养和抚摸的标签。

而实际在功能上被使用的函数包括isMainCharTouch检测是否在主角周围按下Q进行抚摸，decreaseHappiness如果一定时间没有抚摸则幸福度自己下降，move统一启动所有动画。

01 牧场代码实现

```
void Cow::onKeyPressed(EventKeyboard::KeyCode keyCode, Event* event)
{
    if (isNearSprite && keyCode == cocos2d::EventKeyboard::KeyCode::KEY_E && wheat_number > 0)
    {
        if (weather == 1) {
            happiness = happiness / 2; // 减半快乐值
        }
        else if (weather == 2) {
            happiness = happiness / 2; // 减半快乐值
        }
        cow_feed_label->setVisible(true); // 显示文字
        if (happiness <= 50&&happiness>=0)
            milk_number++;
        else if(happiness >= 50&&happiness<=90)
            milk_number+=2;
        else
            milk_number+=4;
        wheat_number--;
        // 启动定时器，3秒后隐藏文字
        this->scheduleOnce(CC_SCHEDULE_SELECTOR(Cow::hideLabel), 3.0f);
    }
    if (isNearSprite && keyCode == cocos2d::EventKeyboard::KeyCode::KEY_Q)
    {
        cow_touch_label->setVisible(true); // 显示文字
        happiness += 10; // 加快乐值
        // 启动定时器，3秒后隐藏文字
        this->scheduleOnce(CC_SCHEDULE_SELECTOR(Cow::hideLabel), 3.0f);
    }
}
```

Cow的核心功能基本由该函数实现，首先会检测是否在动物旁边以及按键E是否被按下，喂养的小麦是否还有剩余，如果以上条件都满足，那么就会进行喂养操纵，喂养时会更新幸福值，如果当前为暴晒或雨天，动物的幸福值都会减半，之后会跳出弹窗提示喂养成功，在检测当前动物的幸福值，如果在0到50给一个牛奶，50到90给两个，超过90给4个，在此之后标签显示3s后消失，抚摸功能基本同理。

02 农场功能

基本种植功能

玩家可种植并收获多种作物，其成熟时间会受到天气影响。有基本的浇水，施肥功能，如未及时浇水作物会干旱而死，施肥可加快成熟速度。

商店功能

玩家可在商店中进行买卖物品，获得农作物，售出农产品，并且某些产品的价格也会随着天气等外部条件而变化。

02 农场功能

作物种植过程

在特定区域锄地后，依次进行播种浇水即可收获作物。其中施肥会加快成熟速度，而未及时浇水作物也会死亡，时间也会受到天气影响。如在干旱天气下，种子会更快死亡。



02 农场功能

市场系统

实现基本的买卖功能。

同时价格也会受到一定的外部波动，如在干旱天气，作物种子的价格会相应提高。

正常情况



不同天气



PART 02

社区交互

- 1.与镇上的居民建立关系，包括友谊和浪漫关系。
- 2.参与定期的社区活动和节日庆典。
- 3.接受并完成居民的委托任务，提高声望和收获奖励，任务包括收集特定物品、帮助修复建筑等

2 社区交互

2.1 居民交互

2.1.1 日常对话（亲密度+10）



2 社区交互

2.1.1 居民对话--接收任务（亲密度+10）



2 社区交互

2.1.1 居民对话--完成任务（亲密度+70）



2 社区交互

2.1.1 居民对话--友谊亲密度

满



2 社区交互

2.1.1 居民对话--恋爱亲密度



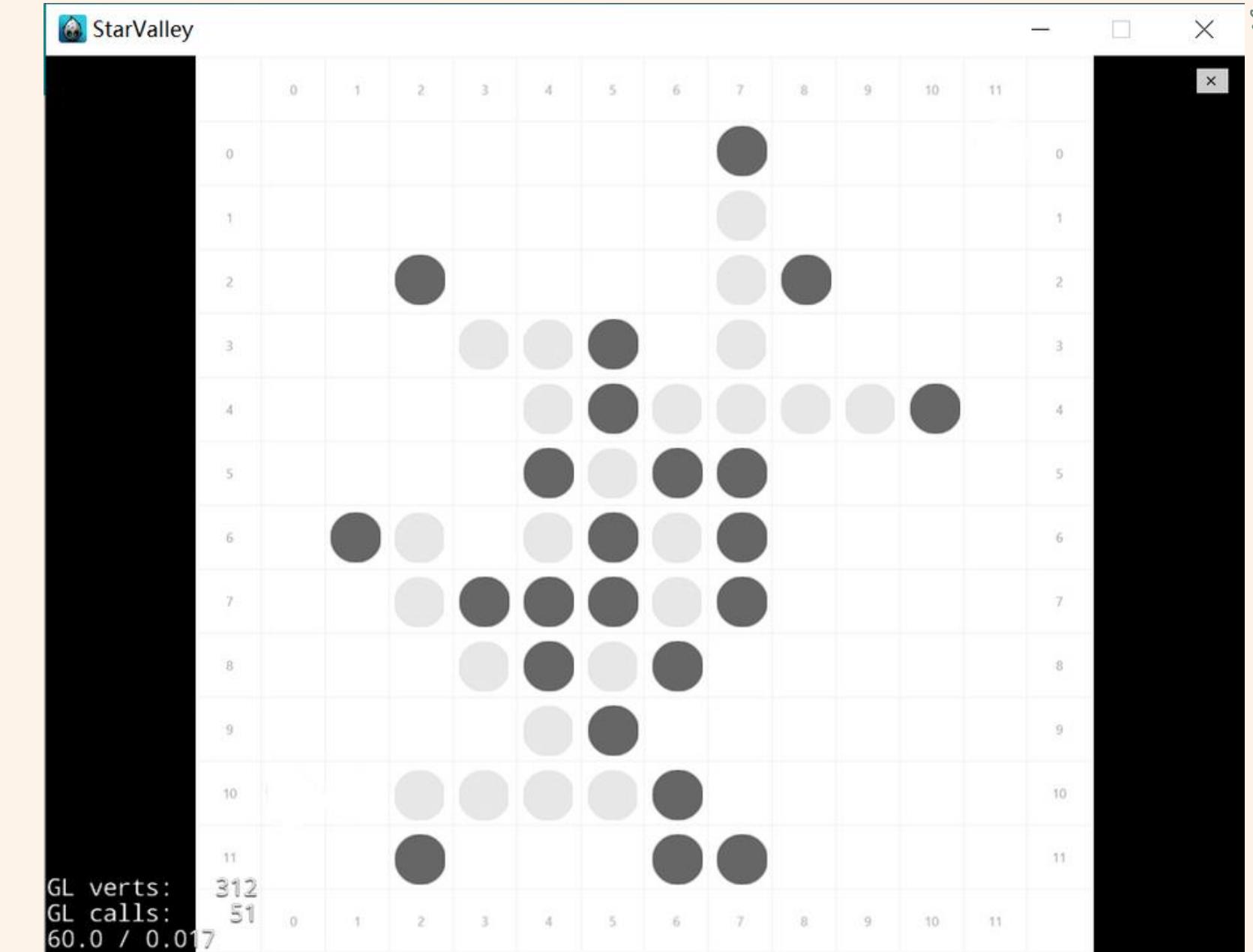
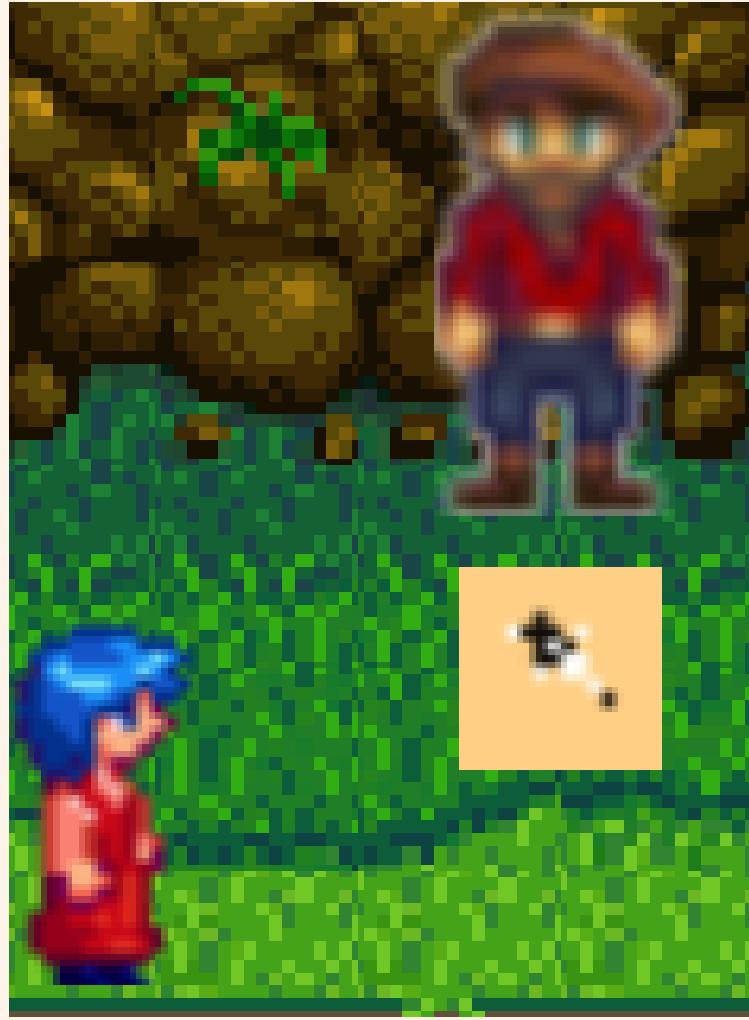
2 社区交互

2.1.2 居民交互-最真实的物理引擎



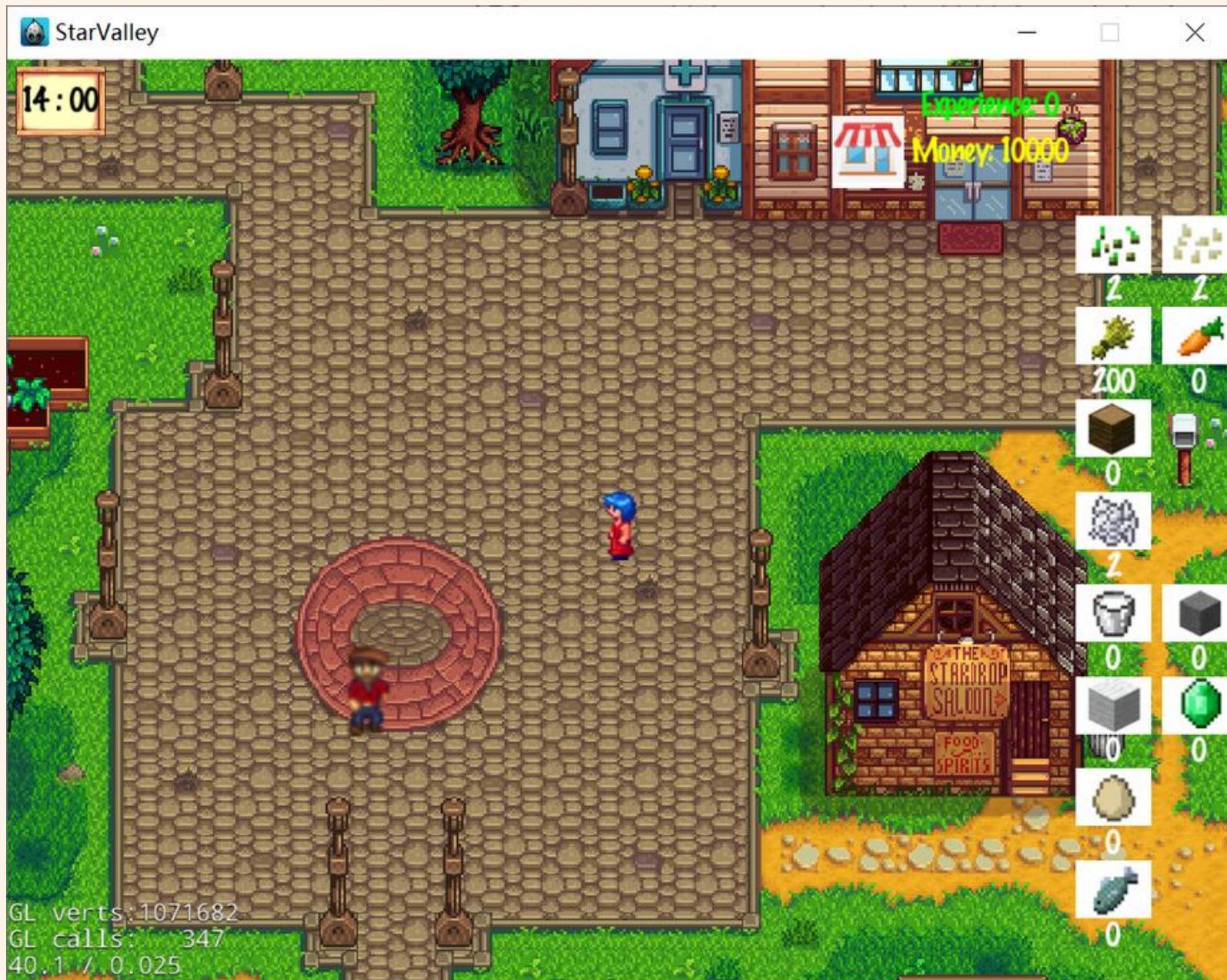
2 社区交互

2.1 五子棋小游戏

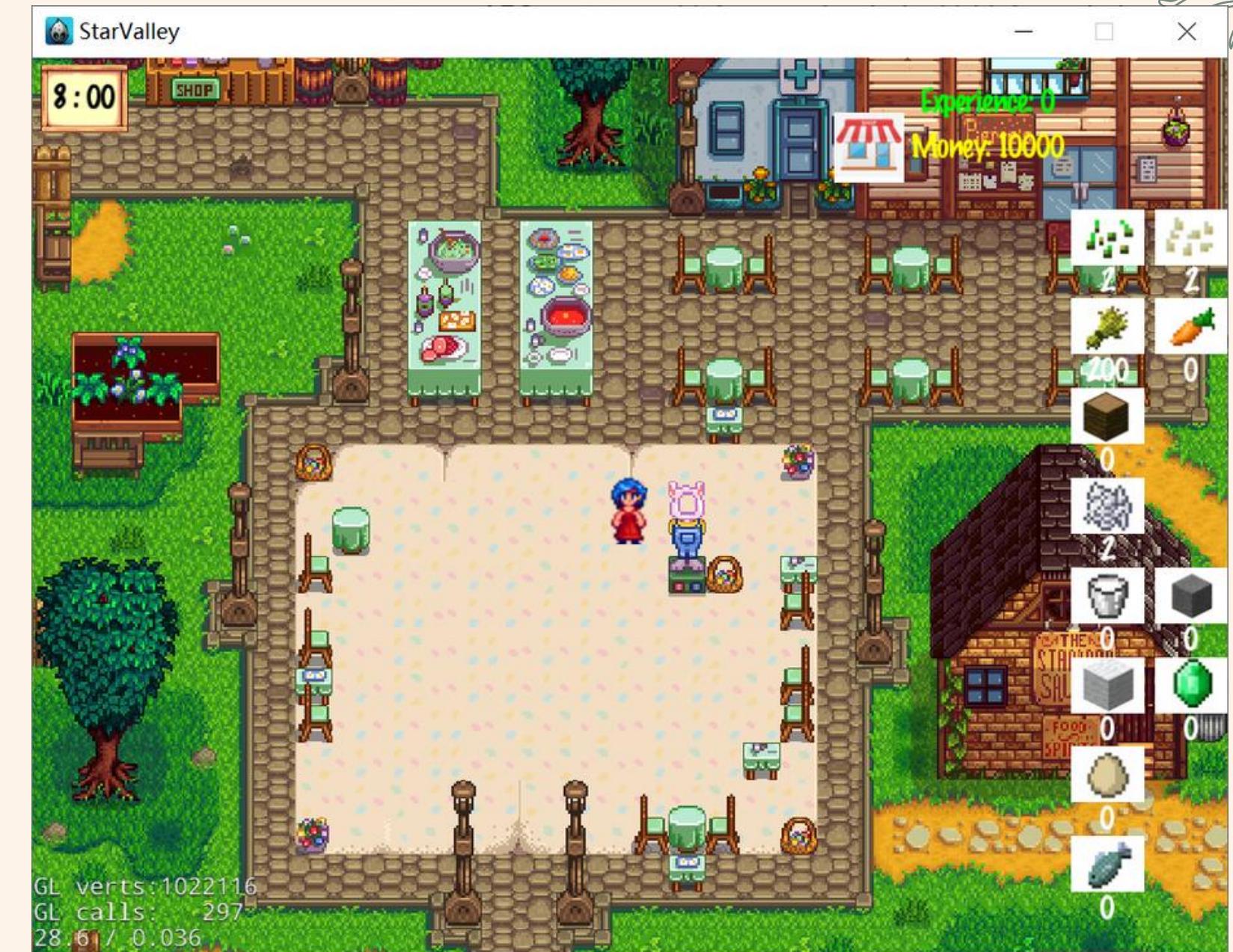


2 社区交互

2.2 定期活动



平时



节日

2 社区交互

2.2.2 昼夜更替和日期时间系统



2 社区交互

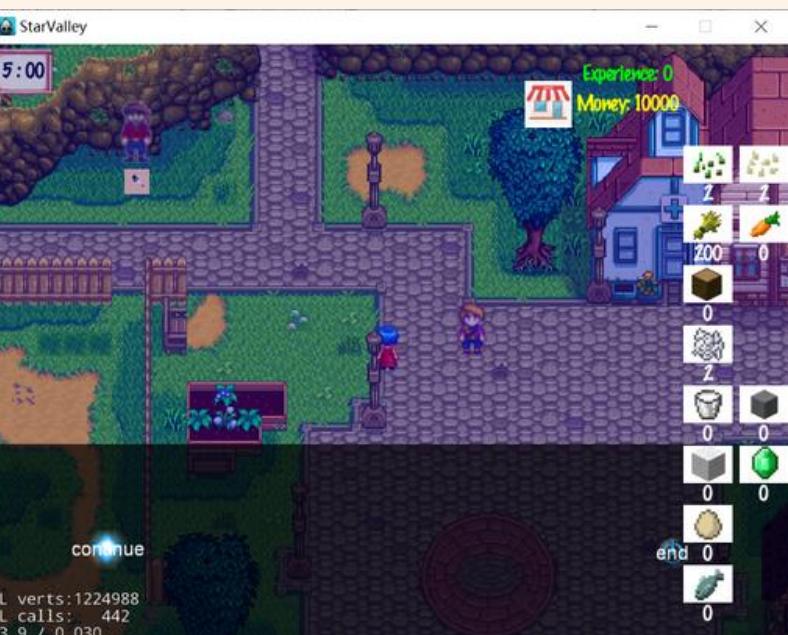
2.3.1 任务的接取

1. 系统分发



2. 去NPC处领

取



2 社区交互

2.3.2 任务的完成



完成后自动从任
务列表中删除



若是NPC任务
则与NPC对话完成
并增加好感度

PART 03

探索和冒险

1. 探索农场周边的地区，包括森林、山脉和神秘洞穴。
2. 挖掘矿物、钓鱼和收集稀有物品。

3.探索和冒险

3.1 探索周边地区

森林



农场



城镇



山脉

矿区



所有地图由游戏解包后
经本小组程序修改
成适配cocos2d引擎的类型

PART 04

角色成长和技能

1. 角色技能树，包括农业、采矿、钓鱼和料理等
2. 随着技能提升，解锁新的作物种类和烹饪食谱。

经验升级与技能解锁

在屏幕右上角有experience的角色经验值，通过完成不同活动如收货作物砍树等可以收获相应的经验值升级。当到达一定经验时，可以解锁不同技能，如挖矿，钓鱼，烹饪等

如图可解锁不同技能



收集矿物

当解锁该技能后，使用镐子工具采集石头，可以获得不同的矿物和相应经验值。



钓鱼

当解锁该技能后，使用钓鱼工具到特定区域，可以进行钓鱼活动。



04 烹饪功能



01 烹饪功能

可以看见，角色要依次经过砍树，钓鱼，种植的工作升级后才会解锁烹饪系统。

而对于烹饪系统的操作来说，首先要获取足够的原料，在功能解锁后按下I按钮即可打开烹饪菜单，如果原料足够按下combine就会合成对应的食品。

01 烹饪功能代码实现

```
lass Ingredients {
public:
    std::string ingredientsName;
    int *quantity;
    std::string imageFile;

    Ingredients(std::string ingreName, int *itemQuantity, std::string itemImageFile)
        : ingredientsName(ingreName), quantity(itemQuantity), imageFile(itemImageFile) {}

    void addQuantity(int qty) {
        *quantity += qty;
    }

    bool consumeQuantity(int qty) {
        if (*quantity >= qty) {
            *quantity -= qty;
            return true;
        }
        return false;
    }
};
```

烹饪功能由两个类共同完成功能第一个是食品原理类，包含了食品的名称，数量，以及资源中图片的名称，包含了构造函数，addQuantity增加数量用的函数和consume消耗的函数。