高级语言程序设计

实验报告

南开大学 人工智能学院

姓名 李云鹏

学号 2414099

班级 0516

2025年5月4日

目录

[高级语言程序设计大作业实验报告 1](#_Toc21286)

[一. 作业题目 1](#_Toc29969)

[二. 开发软件 1](#_Toc18364)

[三. 课题要求 1](#_Toc742)

[四. 主要流程 1](#_Toc25240)

[1． 整体流程 2](#_Toc20452)

[2． 核心算法 2](#_Toc25354)

[（1） 2](#_Toc22508)

[（2） 2](#_Toc28665)

[（3） 3](#_Toc32428)

[（4） 3](#_Toc23860)

[（5） 3](#_Toc13349)

[3．UI特效系统 3](#_Toc24674)

[4．内存管理 3](#_Toc24674)

[五. 单元测试 3](#_Toc28265)

[六. 遇到的问题 4](#_Toc23116)

[七. 收获 4](#_Toc23116)

高级语言程序设计大作业实验报告

1. **作业题目**

基于Qt框架的桌面宠物及跑酷游戏

1. **开发软件**

Qt Creator 15.0.1 (Community)

1. **课题要求**
2. 面向对象设计：采用面向对象编程思想，实现高内聚低耦合的代码结构
3. 单元测试：对核心功能模块进行单元测试验证
4. 模型部分：

桌面宠物：交互式动画角色

跑酷游戏：实现2D无限跑酷游戏机制

1. 验证：通过功能测试确保程序正确性
2. **主要流程**
   1. **整体流程**

先展示桌面宠物系统，采用Qt图形视图框架，实现7种角色动画状态，并通过右键菜单交互系统切换状态，实现可拖动无边框窗口；

然后再展示跑酷游戏系统，采用双背景无限滚动算法，引入真实的物理引擎（如重力、碰撞），并加入道具收集和生命值系统，实现游戏状态管理。

1. **初始化系统**

GameWindow::GameWindow(QWidget \*parent, bool standalone) : QMainWindow(parent)

{

// 窗口设置：无边框、固定大小800x600

// 场景和视图初始化

// 背景图片加载和设置（双缓冲实现无限滚动）

// 玩家角色初始化

// 定时器系统设置（游戏主循环和蛋糕生成）

// UI元素初始化（分数和生命值显示）

}

1. **游戏主循环**

void GameWindow::update()

{

// 背景滚动逻辑（双背景交替）

// 玩家移动处理

// 障碍物生成

// 蛋糕移动和清理

// 碰撞检测

}

* 1. **核心算法**

1. **动画状态显示器**

enum RoleAct{

cold, fly, happy, jump,

liedown, oioioi, sayhello

};

使用Qtimer驱动帧动画更新，并通过枚举管理状态切换；

1. **无限背景滚动**

bgItem1->setX(bgItem1->x() - bgScrollSpeed);

bgItem2->setX(bgItem2->x() - bgScrollSpeed);

if(bgItem1->x() + bgItem1->pixmap().width() <= 0) {

bgItem1->setX(bgItem2->x() + bgItem2->pixmap().width());

}

// 同理处理bgItem2...

通过两个镜像对称的图片循环播放实现背景无限滚动效果；

1. **物理运动模型**

void Player::move(){

velocityY += GRAVITY; // 重力加速度

setPos(x(), y() + velocityY);

if(y() >= GROUND\_LEVEL){ // 地面碰撞检测

setPos(x(), GROUND\_LEVEL);

isJumping = false;

}

}

模拟重力加速度，实现更真实的运动状态；

1. **碰撞检测系统**

void GameWindow::checkCollisions()

{

QList<QGraphicsItem\*> colliding = player->collidingItems();

for (QGraphicsItem\* item : colliding) {

if (Coin \*coin = dynamic\_cast<Coin\*>(item)) {

// 处理蛋糕收集...

}

else if (Obstacle \*obstacle = dynamic\_cast<Obstacle\*>(item)) {

// 处理障碍物碰撞...

if (m\_lives <= 0) {

// 游戏结束处理...

}

}

}

}

1. **动态物品生成系统**

// 障碍物生成

void GameWindow::spawnObstacle()

{

static int frameCounter = 0;

if (++frameCounter >= nextSpawnInterval) {

frameCounter = 0;

nextSpawnInterval = randomGen->bounded(250, 500);

// 创建并放置新障碍物...

}

}

// 蛋糕生成

void GameWindow::spawnCoin()

{

Coin \*coin = new Coin();

coin->setPos(800, randomGen->bounded(200, 400));

// 设置收集信号处理...

}

* 1. **UI特效系统**

**（1）浮动文字效果**

void GameWindow::showFloatingText(const QPointF &position, const QString &text)

{

// 创建文字项

// 设置向上移动动画

// 设置渐隐效果

// 动画完成后自动清理

}

（2）**资源交换系统**

void GameWindow::keyPressEvent(QKeyEvent \*event)

{

if (event->key() == Qt::Key\_Q && coinCount>=10) {

coinCount-=10; // 消耗10个蛋糕

m\_lives++; // 增加1点体力

// 更新UI显示...

}

}

* 1. **内存管理**

GameWindow::~GameWindow()

{

// 清理场景、视图、玩家角色

// 删除所有蛋糕对象

// 停止并删除定时器

}

在析构函数中及时清除堆区数据，防止内存泄漏

1. **单元测试**

| **测试模块** | **测试用例** | **预期结果** | **实际结果** |
| --- | --- | --- | --- |
| 角色移动 | 跳跃指令 | 垂直速度变化 | ✔ |
| 碰撞检测 | 角色与金币接触 | 分数增加 | ✔ |
| 背景滚动 | 300帧运行 | 无缝循环 | ✔ |
| 状态切换 | 点击不同菜单项 | 正确动画切换 | ✔ |

1. **遇到的问题**
2. **角色穿透地面**

**这是因为没有加入落地检测，导致下落一直执行**

if (y() + pixmap().height() >= groundLevel) {

setPos(x(), groundLevel - pixmap().height());

// ...

}

1. **角色与背景位置不匹配**

即角色与背景的坐标系统没有统一。应先设定一个统一的地面高度GROUND\_Y=500,再根据此数值修改角色的初始化部分

// 在GameWindow构造函数中修改：

const int GROUND\_Y = 500; // 统一定义地面高度

// 玩家设置

player = new Player();

scene->addItem(player);

player->setPos(100, GROUND\_Y - player->pixmap().height());

// 角色底部接触地面

player->setZValue(100);

1. **按下W后角色没有起跳**

即键盘事件没有被正确捕获，应修改player.cpp，使角色获得焦点

Player::Player() : velocityY(0), isJumping(false)

{

// ...动画初始化代码...

// 关键设置：允许接收键盘事件

setFlag(QGraphicsItem::ItemIsFocusable);

setFocus();

}

void Player::keyPressEvent(QKeyEvent \*event)

{

if (event->key() == Qt::Key\_Space && !isJumping) {

isJumping = true;

velocityY = -15;

animationTimer->setInterval(50);

// 调试输出

qDebug() << "Jump triggered!";

}

QGraphicsPixmapItem:keyPressEvent(event); // 传递事件

}

1. **进入游戏时，角色跑步动画保持播放一帧**

我在构造函数初始化列表里设置了currentState=Running，同时启动了animationTimer，但是游戏一开始角色只显示同一帧，只有按下W才开始播放跑步动作。

但是定时器并没有被触发。本类继承了QGraphicsPixmapItem和QObject，但是我把antimationTimer的父对象写成了this,这个this是QGraphicsItem,不在主线程的eventloop里。

解决办法是使用QTimer::singleShot，它与主线程无关，每次antimate()执行完自动安排下一次调用。

void Player::animate()

{

currentFrame++;

switch (currentState) {

case Running:

if (!runFrames.isEmpty()) {

setPixmap(runFrames[currentFrame % runFrames.size()]);

update();

}

break;

case Ducking:

if (!duckFrames.isEmpty()) {

setPixmap(duckFrames[currentFrame % duckFrames.size()]);

update();

}

break;

case Jumping:

if (!runFrames.isEmpty()) {

setPixmap(runFrames[currentFrame % runFrames.size()]);

update();

}

break;

}

// 自己驱动下一帧

QTimer::singleShot(animationTimer->interval(), this, &Player::animate);

}

1. **收获**
2. 大致掌握了Qt的基本框架，搭建了完整功能的图形化程序结构；学会使用了QGraphicsScene,Qtimer,QPinmap等组件；

成功处理了图像、碰撞检测、动画帧更新等图形相关技术。

1. 大致掌握了C++的项目管理功能，学会了使用CMake构建项目，添加资源文件，链接模块；

初步了解模块依赖和跨平台配置方法

3.基本掌握了面向对象编程的思想，把不同对象分成不同的类，分别进行逻辑实现