南开大学 计算机大类

杨紫辰

2313227

网络空间安全模拟2.1班

2024年5月5日

高级语言程序设计实验报告

1. 作业题目

Go Elysia！

创作缘由：源于对米哈游游戏公司旗下崩坏三这款游戏中的人物角色爱莉希雅的热爱，并且受到童年游玩游戏《天天酷跑》的启发，初步想法是做一个跑酷类型的游戏，在后续又慢慢有了这样一个游戏的架构。

1. 开发软件

Qt 5.99（Qt Creator 4.11.0）

1. 课题要求
2. 练习类与继承
3. 学习掌握Qt基本操作
4. 能够自主完成一个项目
5. 能够讲解自己的项目
6. 主要流程
7. 搭建进入游戏主界面

实现MainWindow类

现在介绍其中关键的几个函数

void **Init**(); // 初始化界面函数

void ***paintEvent***(QPaintEvent \*event); // 绘制事件

void **showMainWindow**(); // 显示主窗口

void **onExitClicked**(); // 退出按钮点击事件

void **onStaClicked**(); // 开始按钮点击事件

void **onIntroClicked**(); // 介绍按钮点击事件

void **onPriClicked**(); // 成就按钮点击事件

其中在Init函数中，我们完成了主界面的布置，主要包括一些按钮的定义和美化，以及主界面的PixMap的设置。

而四个按钮点击事件则分别对应了点击按钮时所释放的信号和对应操作（大多涉及界面的切换），而退出按钮则是为消息框的弹出。

1. 人物的创建

实现aili类

现在介绍其中一些关键的函数

QRect aili\_Rect; // Aili 对象的矩形区域

QPixmap aili\_pixmap; // Aili 对象的图片

void **jump**(); // 跳跃动作

void **rush**(); // 冲刺动作

void **trick**(); // 技能动作

void **rise**(); // 上升动作

void **fall**(); // 下落动作

QPixmap **getPixmap**() const; // 获取 Aili 对象的图片

int **getX**() const; // 获取 Aili 对象的 X 坐标

int **getY**() const; // 获取 Aili 对象的 Y 坐标

void **setPosition**(int x, int y); // 设置 Aili 对象的位置

bool isJumping; // 是否正在跳跃

bool isRushing; // 是否正在冲刺

bool isTricking; // 是否正在使用技能

之所以将aili设置为一个矩形，是为了后续方便进行碰撞检测（最开始的设想其实是设置成为Label）。

Jump、rush、trick分别对应aili的三个不同的动作，jump实现跳跃；rush实现俯冲、trick实现大招的释放。Jump主要涉及矩形高度变化，随着定时器时间增多速度变快或变慢（上升变慢，下降变快）；rush调整矩形高度，改变移动速度；trick实现上升并且无敌效果，改变pixmap的赋予。

剩下的函数就是一些状态的获取或判断。

1. 地面的滚动实现

class **Ground**

class **Grounds**

由两个类进行管理，Ground中对单一的地面设置滚动速度和位置移动实现。Grounds则是一个储存Ground的数组，设定恰当的移动速度而实现地面的连贯性和滚动性。

1. 障碍物的定义

这里我们主要讲一下基类的组成，子类都是在基类基础上进行虚函数的实现。

virtual void ***updatePosition***() = 0; // 更新障碍物位置的纯虚函数

virtual int ***ifCollision***(QRect) = 0; // 检查是否碰撞的纯虚函数

virtual bool ***ifGoOut***() = 0; // 检查障碍物是否移出屏幕的纯虚函数

virtual QPixmap ***getPixmap***() = 0; // 获取障碍物图片的纯虚函数

int w, h; // 障碍物的宽度和高度

int speed; // 障碍物的速度

mutable QPixmap img; // 障碍物图片（mutable 修饰，因为在 const 成员函数中可能需要修改）

QRect barrier; // 障碍物的矩形边界

更新位置即按照滚动速度每隔一定时间更新障碍物位置，实现滚动效果；检测碰撞是用于判断是否与aili相撞；检测障碍物是否移出屏幕只用判断障碍物的x坐标是否出界，在后续根据此来删除出界的障碍物，剩下为一些基础设置。

1. 关卡的设置

first\_cham类的实现

QVector <barriers\*>barriers; // 管理屏幕中的障碍物的向量

void **updateGround**(); // 更新地面状态

void **updatebarriers**(); // 更新障碍物状态

void **storeBarriers**(); // 存储障碍物状态

void **gameOver**(); // 游戏结束处理函数

void **gameStart**(); // 游戏开始处理函数

void **ifCollision**(); // 碰撞检测函数

void **showRestartDialog**(QWidget \*parent); // 显示重新开始对话框

值得讲的部分就是障碍物的存储，这里使用了向量，更为方便的动态管理障碍物，如果出了屏幕就删除，满足一定的条件（如时间），便能向内存储障碍物，使屏幕上不断刷出新的障碍物。更新地面和障碍物就是对他们坐标的改变，store就是往向量内存储障碍物，在游戏开始和结束函数内，我们改变一些变量的状态，使游戏从头开始或停止。

1. 美化补充

如补充游戏介绍等功能，美化Pixmap，使图片高清化或者改变图片的朝向，使游戏更加自然。

1. 一些可以改进的地方
2. 成就面板的实现，目前只是将其内容搭建完成，功能并未完全实现，有些遗憾。详细见prize类
3. 游戏关卡设置，原先是想设计多关，但时间和精力不足，最终只实现了一关。
4. 本来想实现文本对话，但简单尝试了一下，不知道怎么美化，最后放弃了，残留了一部分见dialogue类