高级语言程序设计大作业实验报告

一. 作业题目

使用 C++语言完成一个图形化的小程序

此处选择用 QT 作为图形界面应用程序框架，实现躲避障碍物前进的小游戏

二. 开发软件

QT6.5.3

三. 课题要求

自行设计 QT 程序，实现功能基本完整的小程序

四. 主要流程

1．类的设计

实现思路:

定义 config 文件用于存放游戏的配置信息，比如在里面定义了窗口的长和宽，窗口的标题等内容

创建 mainscene 类作为主场景，里面有启动界面和游戏界面，包括控制游戏开始的按钮，播放背景音乐和停止背景音乐的按钮，在 mainscene 中定义了一个定时器并设置间隔用于游戏中元素坐标的更新，碰撞检测，障碍物出场的功能和鼠标事件的检测

Mainscece 中包含了玩家操纵的角色一个类，障碍物的一个类和地图的一个类

各自都加载了相对应的资源文件。在地图类里面在对应位置插入了两个图片，通过变化他们的位置来实现背景无限滚动的效果。在障碍物类呢设置了一个数组，对障碍物加载了不同的资源图片实现障碍物的多样化，以及障碍物的横坐标，纵坐标，矩形框和更新位置的一个函数。在角色类内也是设置了横坐标，纵坐标，矩形框（用于碰撞检测），通过设置鼠标移动事件来对角色进行移动来躲避掉落下来的障碍物，下图为鼠标移动事件：

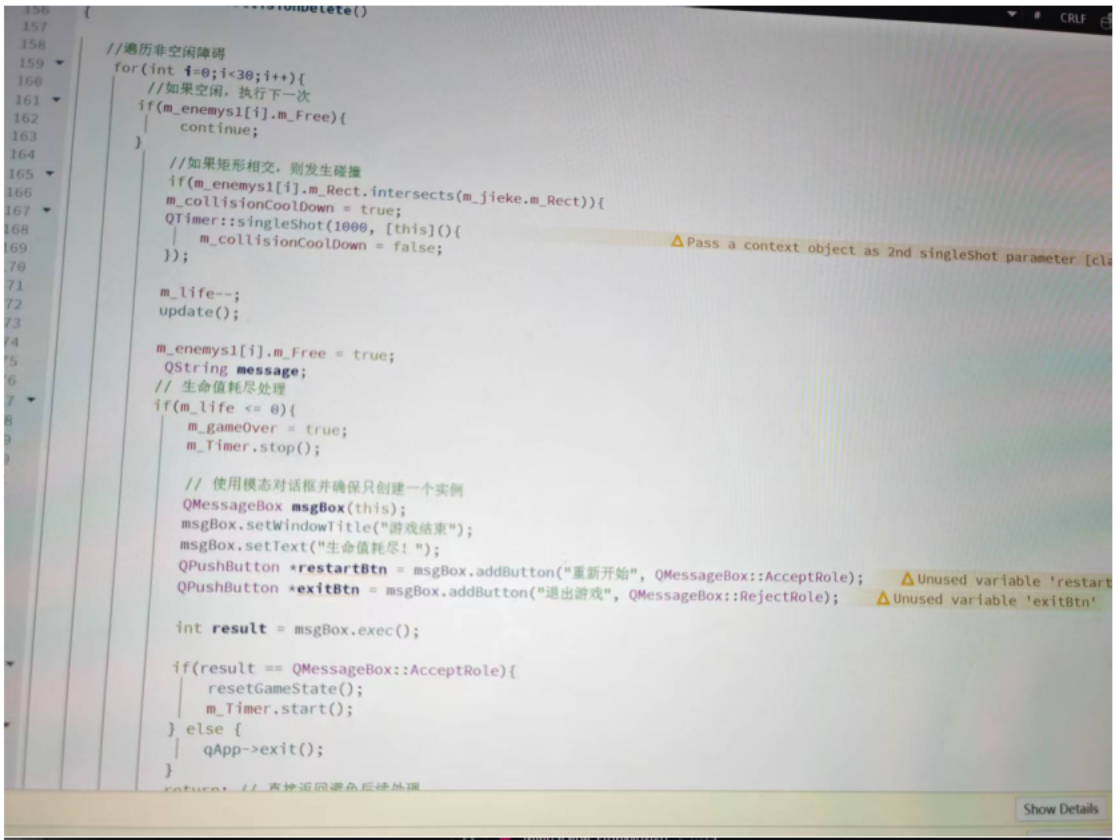
```
void MainScene::mouseMoveEvent(QMouseEvent *event)
{
    if(m_gameOver) return;
    int x=event->x()-m_jieke.m_Rect.width()*0.5;
    int y=event->y()-m_jieke.m_Rect.height()*0.5;
    //边界检测
    if(x<=0){
        x=0;
    }
    if(x>=GAME_WIDTH-m_jieke.m_Rect.width()){
        x=GAME_WIDTH-m_jieke.m_Rect.width();
    }
    if(y<=0){
        y=0;
    }
    if(y>=GAME_HEIGHT-m_jieke.m_Rect.height()){
        y=GAME_HEIGHT-m_jieke.m_Rect.height();
    }
    m_jieke.setPosition(x,y);
}
```

还有播放音乐和停止音乐的函数，通过 connect 与按钮的点击联系起来从而实现音乐播放和停止的功能

核心代码（碰撞检测）

思路：首先遍历非空闲的障碍物，判断障碍物的矩形框与角色自身的矩形框如果相交则说明二者发生碰撞，然后将障碍物的 m.Free 状态设为 true，这样让障碍

物消失并设置对话框在碰撞的时候弹出来

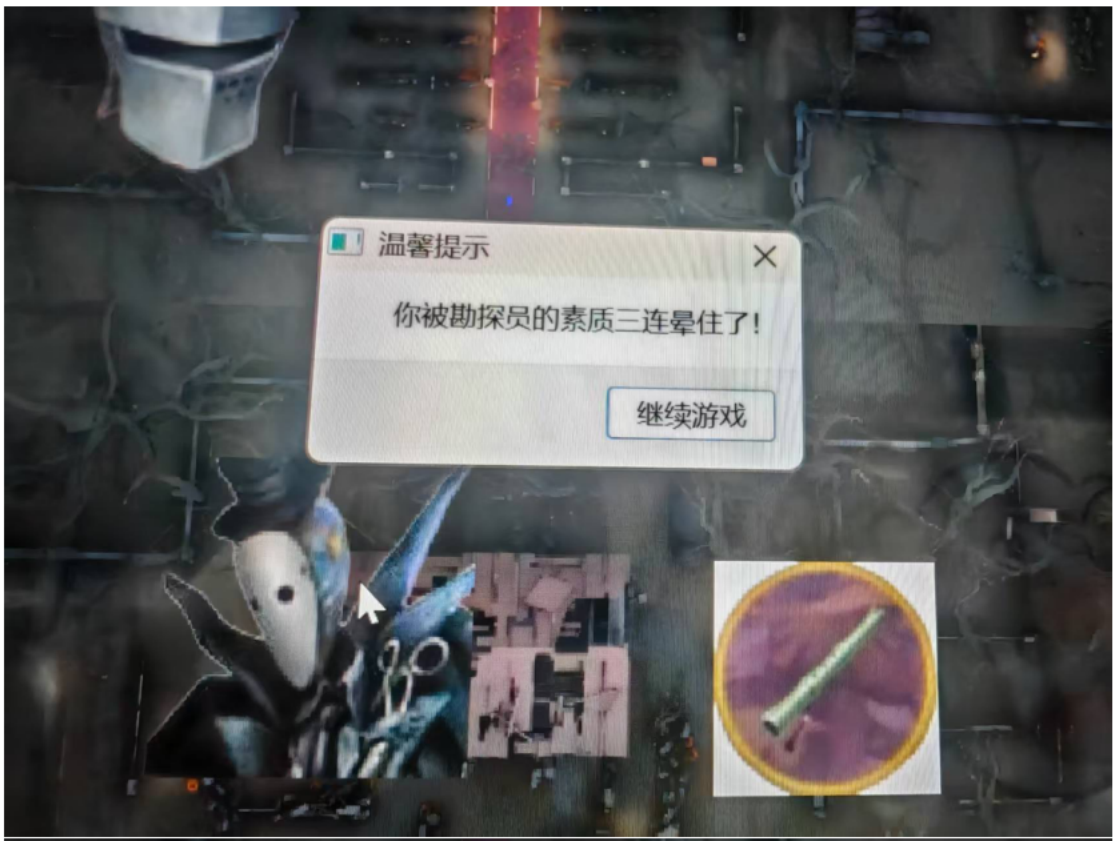


```
//遍历非空闲障碍
for(int i=0;i<30;i++){
    //如果空闲，执行下一次
    if(m_enemys1[i].m_Free){
        continue;
    }
    //如果矩形相交，则发生碰撞
    if(m_enemys1[i].m_Rect.intersects(m_jieke.m_Rect)){
    m_collisionCoolDown = true;
    QTimer::singleShot(1000, [this](){
        m_collisionCoolDown = false;
    });

    m_life--;
    update();

    m_enemys1[i].m_Free = true;
     QString message;
    // 生命值耗尽处理
    if(m_life <= 0){
        m_gameOver = true;
        m_Timer.stop();

        // 使用模态对话框并确保只创建一个实例
        QMessageBox msgBox(this);
        msgBox.setWindowTitle("游戏结束");
        msgBox.setText("生命值耗尽！");
        QPushButton *restartBtn = msgBox.addButton("重新开始", QMessageBox::AcceptRole);
        QPushButton *exitBtn = msgBox.addButton("退出游戏", QMessageBox::RejectRole);

        int result = msgBox.exec();

        if(result == QMessageBox::AcceptRole){
            resetGameState();
            m_Timer.start();
        } else {
            qApp->exit();
        }
```

```
            else{
            m_enemys1[i].m_Free=true;
            switch (i%6) {
            case 0:
                message = "你被前锋的超长螺旋球撞晕了，请眩晕10秒钟！";
                break;
            case 1:
                message = "你被古董商的三连棍打中啦!18秒内不能攻击！";
                break;
            case 2:
                message = "你被勘探员的素质三连晕住了！";
                break;
            case 3:
                message = "你被骑士猜了普攻！你将无法使用普通攻击！";
                break;
            case 4:
                message = "你被空军的信号枪击中了！请眩晕5.5秒！";
                break;
            case 5:
                message = "你被火箭撞上了！请原地罚站！";
                break;
            default:
                break;
            }

        }

        // 创建对话框实例
        QMessageBox msgBox;
        msgBox.setWindowTitle("温馨提示");  // 标题
        msgBox.setText(message);           // 内容

        // 添加按钮
        QPushButton *customOkBtn = msgBox.addButton("继续游戏", QMessageBox::AcceptRole);
        msgBox.setDefaultButton(customOkBtn);  // 设置为默认按钮

        // 显示对话框
        msgBox.exec();
        }
    }
```

五. 单元测试

测试碰撞检测函数
当角色自身与不同的障碍物碰撞的时候会弹出不同的对话框
下为两个简单的测试



六. 收获

1. config 配置文件的作用

在 config 文件中集中对一些常量进行了配置。

好处是如果程序有很多常量我们需要改正，我们只需要改动 config 文件中的数据即可，不需要改动我们程序的主体，以免改动程序错误出现 bug。

2. QT 帮助文档的使用

在遇到一些不明确的错误时可以去 QT 的帮助文档里面搜索相关的函数看看对应的语法，或者想知道该对象还具有什么功能函数时，也可以查阅 QT 的帮助文档进行了解。

3. 析构函数的使用

当我们在堆区手动创建了一些数据并且未将他们放到对象树上时，此时需要我们自己在析构函数中手动释放它们，以免照成内存泄漏等问题。