高级语言程序设计

实验报告

南开大学 计算机大类

姓名 尹天宇

学号 2414063

班级 计算机卓越班

2025 年 5 月 15 日

一、作业题目

本次作业实现一个基于 Qt 框架的射击游戏，玩家通过鼠标控制英雄移动并发射子弹，射击从屏幕右侧随机出现的敌人，在规定时间内达到目标分数即可获胜。

二、开发软件

本项目使用 Qt 进行开发，Qt 是一个跨平台的 C++ 应用程序开发框架，提供了丰富的图形界面组件和工具，方便开发者快速创建功能强大的应用程序。

三、课题要求

1. **游戏界面**：设计一个美观、易用的游戏界面，包含游戏场景、英雄、敌人、子弹等元素。
2. **游戏逻辑**：实现英雄的移动、子弹的发射、敌人的出场和移动、碰撞检测等基本游戏逻辑。
3. **游戏音效**：添加游戏背景音乐和碰撞音效，增强游戏的趣味性。
4. **游戏计时和计分**：设置游戏时间限制，在规定时间内达到目标分数则游戏胜利，否则游戏失败。
5. **用户交互**：支持鼠标控制英雄的移动，方便玩家操作。

四、主要流程

1. 整体流程

本游戏的整体流程可以分为以下几个步骤：

- **主窗口显示**：程序启动后，显示主窗口，主窗口包含“开始游戏”、“播放音乐”按钮和音量调节滑块。

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    //标题
    setWindowTitle("我不是笨蛋奶妈");

    // 设置窗口大小
    resize(1868, 1041);

    // 设置背景图片
    QPalette palette;
    QPixmap pixmap(":/images/image1.png");
    palette.setBrush(QPalette::Window, QBrush(pixmap.scaled(this->size(), Qt::IgnoreAspectRatio, Qt::SmoothTransformation)));
    this->setPalette(palette);

    //按钮
    button = new QPushButton("开始游戏", this);
    button->setGeometry(100, 100, 200, 50);
    connect(button, &QPushButton::clicked, this, &MainWindow::onButtonClicked);

    // 音乐开关按钮
    musicButton = new QPushButton("播放音乐", this);
    musicButton->setGeometry(100, 160, 200, 50);
    connect(musicButton, &QPushButton::clicked, this, &MainWindow::onMusicButtonClicked);

    // 音量调节拖动条
    volumeSlider = new QSlider(Qt::Horizontal, this);
    volumeSlider->setGeometry(100, 220, 200, 20);
    volumeSlider->setRange(0, 100);
    volumeSlider->setValue(50);
    connect(volumeSlider, &QSlider::valueChanged, this, &MainWindow::onVolumeSliderValueChanged);

    // 初始化音效
    soundEffect = new QSoundEffect(this);
    soundEffect->setSource(QUrl::fromLocalFile(MUSIC_PATH));
    soundEffect->setVolume(0.5);
    soundEffect->setLoopCount(QSoundEffect::Infinite); // 循环播放

    mainScene = new MainScene();
}
```

- **游戏初始化**：点击“开始游戏”按钮后，隐藏主窗口，显示游戏场景，并初始化游戏元素，如地图、英雄、敌人、子弹等。

```
void MainWindow::onButtonClicked()
{
    this->hide();
    mainScene->show();
    mainScene->initScene();
    mainScene->playGame();
}
```

```cpp
void MainScene::initScene()
{
    //设置窗口尺寸
    setFixedSize(GAME_WIDTH,GAME_HEIGHT);

    //标题
    setWindowTitle(GAME_TITLE);

    //定时器
    m_Timer.setInterval(GAME_RATE);

    //初始化enemy出场时间间隔
    m_recorder=0;

    //随机数种子
    srand((unsigned int)time(NULL));

    // 初始化分数
    m_score = 0;

    // 初始化游戏剩余时间
    m_remainingTime = GAMETIME;

    // 启动游戏时间定时器
    m_gameTimer.setInterval(1000); // 每秒更新一次
    connect(&m_gameTimer, &QTimer::timeout, this, &MainScene::updateGameTime);
    m_gameTimer.start();

}
```

- **游戏循环**：启动定时器，在定时器的回调函数中不断更新游戏元素的位置、检测碰撞、绘制游戏界面，直到游戏时间结束。

```
void MainScene::playGame()
{
    //启动定时器
    m_Timer.start();

    //发送信号
    connect(&m_Timer,&QTimer::timeout,[=]()
            {

            if (m_remainingTime > 0) // 游戏时间结束时like停止
            {
                //enemy出场
                enemyToScene();

                //更新坐标
                updatePosition();

                //绘制
                update();

                //碰撞检测
                collisionDetection();
            }
    });
}
```

- **碰撞检测**：在游戏循环中，检测子弹和敌人是否发生碰撞，如果发生碰撞，则将敌人和子弹标记为空闲状态，并增加分数。

```
void MainScene::collisionDetection()
{
    //bianli非空闲enemy
    for(int i=0;i<ENEMY_NUM;i++)
    {
        if( m_enemys[i].m_Free)
        {
            continue;
        }


        //遍历非空闲子弹
        for(int j=0;j<NUMBER;j++)
        {
            //空闲子弹continue
            if(m_hero.m_bullets[j].m_Free || m_enemys[i].m_X >= GAME_WIDTH)
            {
                continue;
            }

            //相交则消失
            if(m_enemys[i].m_Rect.intersects(m_hero.m_bullets[j].m_Rect))
            {
                m_enemys[i].m_Free=true;
                m_hero.m_bullets[j].m_Free=true;
                m_score += SCORE_PER_ENEMY; // 计分

                // 播放碰撞音效
                if (m_collisionCount < 5) {
                    m_collisionSounds[m_collisionCount].play();
                } else {
                    m_collisionSounds[4].play();
                }

                // 增加碰撞计数器
                m_collisionCount++;
            }
        }
    }
}
```

- **游戏结束**：当游戏时间结束时，停止定时器，显示游戏结束对话框，根据玩家的分数判断游戏胜负。

```
// 游戏结束处理
void MainScene::gameOver()
{
    QString message = QString("游戏结束，您的分数为 %1").arg(m_score);
    if (m_score >= TARGET_SCORE)
    {
        message += ".......你的使命还没结束!!!";

    }
    else
    {
        message += "\n当心眼睛!!!";

    }
    newmessagebox msgBox("游戏结束", message,m_score, this);
    msgBox.exec();


    emit gameOverSignal();
}
```

2. **算法或公式**

- **碰撞检测算法**：使用 QRect::intersects()函数检测子弹和敌人的矩形边框是否相交，如果相交则认为发生了碰撞。代码示例如下：

```
//相交则消失
if(m_enemys[i].m_Rect.intersects(m_hero.m_bullets[j].m_Rect))
{
    m_enemys[i].m_Free=true;
    m_hero.m_bullets[j].m_Free=true;
    m_score += SCORE_PER_ENEMY; // 计分
```

- **敌人出场算法**：使用一个计数器记录敌人出场的时间间隔，当计数器达到规定的时间间隔时，从敌人数组中选择一个空闲的敌人，将其设置为非空闲状态，并随机设置其出场位置。代码示例如下：

```cpp
void MainScene::enemyToScene()
{
    //累计时间间隔记录变量
    m_recorder++;

    //未到时间间隔不出场
    if(m_recorder<ENEMY_INTERVAL)
    {
        return;
    }

    //达到时间间隔,出场
    m_recorder=0;

    for(int i=0;i<ENEMY_NUM;i++)
    {
        //空闲出场
        if( m_enemys[i].m_Free)
        {
            m_enemys[i].m_Free = false;

            //设置坐标
            m_enemys[i].m_X=GAME_WIDTH;
            m_enemys[i].m_Y=rand()%(GAME_HEIGHT-m_enemys[i].m_Rect.height());
            break;
        }
    }
}
```

3.核心类结构

```
├─── GameObject (抽象基类)
│      ├─── m_X, m_Y (位置)
│      ├─── m_Rect (碰撞矩形)
│      ├─── m_Image (图像)
│      └─── updatePosition() (纯虚函数)
│
├─── Hero : public GameObject
│      ├─── m_bullets[] (子弹数组)
│      └─── shoot() (发射子弹)
│
├─── Enemy : public GameObject
│      ├─── m_Speed (移动速度)
```

```
|       └──── m_Free (是否空闲)
|
└──── Bullet : public GameObject
      ├──── m_Speed (移动速度)
      └──── m_Free (是否空闲)
```

五、收获

通过本次大作业，我对 Qt 框架有了更深入的了解，掌握了使用 Qt 进行图形界面开发的基本方法，包括窗口的创建、控件的使用、信号与槽的连接等。同时，我也学会了如何实现游戏的基本逻辑，如碰撞检测、定时更新等。在编写代码的过程中，我遇到了一些问题，如内存泄漏、逻辑错误等，通过调试和查阅资料，我成功解决了这些问题，提高了自己的编程能力和解决问题的能力。此外，我还学会了如何进行单元测试，通过单元测试可以及时发现代码中的问题，提高代码的质量和可靠性。总之，本次大作业让我在实践中积累了宝贵的经验，为今后的学习和工作打下了坚实的基础。