南开大学 计算机大类

姓名 胡楚骅

学号 2413276

班级 工科试验班3-3班

2025年5月3日

高级语言程序设计

实验报告

目录

[高级语言程序设计大作业实验报告 2](#_Toc8100)

[一. 作业题目 2](#_Toc13786)

[二. 开发软件 2](#_Toc2808)

[三. 课题要求 2](#_Toc16201)

[四. 主要流程 3](#_Toc22154)

[1． 整体流程 3](#_Toc18081)

[2． 算法或公式 3](#_Toc7547)

[（1）玩家移动边界判断 3](#_Toc2426)

[（2）碰撞检测算法 5](#_Toc15056)

[（3）倒计时与生命周期管理 5](#_Toc32446)

[3.单元测试 5](#_Toc22328)

[五. 单元测试 5](#_Toc27771)

[测试结果 6](#_Toc12116)

[六. 收获 6](#_Toc6288)

[1. 对C++编程语法的进一步熟悉 6](#_Toc23157)

[2.单元测试对于编程十分重要 6](#_Toc18494)

[3.定时器与信号槽的管理...............................................................................................................6](#_Toc7185)

[4.图形效果的实现 6](#_Toc7185)

高级语言程序设计大作业实验报告

1. **作业题目**

躲避激光小游戏在qt图形化平台上的实现

1. **开发软件**

Qt Creator 15.0.1

Qt版本为6.5.3

1. **课题要求**
2. 面向对象。
3. 单元测试。
4. 模型部分
5. 验证
6. **主要流程**
   1. **整体流程**

实现思路：

利用 Qt 框架构建图形界面和逻辑系统。首先，定义了Player、Laser和WarningLaser类，分别代表玩家、激光和警告激光，为游戏元素赋予属性和行为。接着，创建GameView类来管理游戏场景和逻辑，包含生成激光的spawnTimer和检测碰撞的collisionTimer。在main函数中，搭建游戏启动窗口和主场景，设置背景、圆形区域、玩家、计分和倒计时标签等。玩家点击开始按钮后，游戏启动，spawnTimer定时生成警告激光，倒计时结束后转变为实际激光，collisionTimer实时检测玩家与激光的碰撞，若发生碰撞则游戏结束，玩家可按 R 键重新开始，按 Q 键退出游戏。

* 1. **算法或公式**

1. 玩家移动边界判断

玩家在一个圆形区域内移动，为确保玩家不会移出该圆形区域，每次移动时需进行边界判断。具体公式如下：

距离公式：计算玩家移动后的位置与圆心的距离 d，公式为 d = √((x - x₀)² + (y - y₀)²)，其中 (x, y) 是玩家移动后的位置，(x₀, y₀) 是圆心的位置。

边界判断：如果 d ≤ r（r 为圆的半径），则允许玩家移动到该位置；否则，禁止移动。

2. 激光生成算法

2.1 激光起点计算

激光的起点严格位于圆周上，通过以下公式计算：

极坐标转直角坐标：已知圆心坐标 (x₀, y₀)、半径 r 和角度 θ，则起点坐标 (x, y) 为 x = x₀ + r cos (θ)，y = y₀ + r sin (θ)，其中 θ 是激光的起始角度，通过 θ = i × (360°/n) 计算，i 是激光的序号，n 是当前波次生成的激光数量。

2.2 激光方向计算

激光的方向指向圆心附近，在 ±45° 范围内随机选择：

计算相反角度：θ\_opposite = θ + 180°，并对 360° 取模，确保角度在 [0°, 360°) 范围内。

随机选择方向角度：θ\_direction = θ\_opposite - 45° + rand (0, 90°)，同样对 360° 取模。

计算方向向量：dx = cos (θ\_direction)，dy = sin (θ\_direction)，并对向量进行归一化处理，即 dx = dx / √(dx² + dy²)，dy = dy / √(dx² + dy²)。

2.3 激光终点计算

激光终点是激光线与圆周的交点，通过求解直线与圆的方程组得到：

直线方程：x = x\_start + t × dx，y = y\_start + t × dy，其中 (x\_start, y\_start) 是激光起点，(dx, dy) 是方向向量，t 是参数。

圆方程：(x - x₀)² + (y - y₀)² = r²，将直线方程代入圆方程得到关于 t 的一元二次方程：at² + bt + c = 0，其中 a = dx² + dy²，b = 2 (dx (x\_start - x₀) + dy (y\_start - y₀))，c = (x\_start - x₀)² + (y\_start - y₀)² - r²。

求解 t：使用求根公式 t = (-b ± √(b² - 4ac)) / (2a)，选择远离起点的交点（通常为较大的 t 值）。

计算终点坐标：x\_end = x\_start + t × dx，y\_end = y\_start + t × dy。

3. 碰撞检测算法

通过检查玩家是否与激光发生碰撞来判断游戏是否结束，使用 QGraphicsItem::collidesWithItem 函数进行碰撞检测。

4. 倒计时与生命周期管理

警告激光倒计时：使用 QTimer 实现倒计时功能，每隔 1 秒更新剩余时间，当倒计时结束时发射 countdownFinished 信号。

激光生命周期管理：激光有一个固定的生命周期，使用 QTimer 实现，当生命周期结束时发射 lifetimeFinished 信号，并将激光从场景中移除。

* 1. **单元测试**

在源代码当中添加测试代码并运行，在运行过程中进行对应操作并检查应用程序输出。如果软件执行了对应操作并且应用程序输出窗口可以正常输出测试成功预设内容，说明测试通过。

1. **单元测试**

表 1：测试案例

|  |  |  |
| --- | --- | --- |
| **输入** | **输出** | **目的** |
| 按方向键 | 角色移动并且正常输出测试成功 | 检测角色移动是否正常 |
| 在游戏结束时按R键 | 游戏正常重新开始并且正常输出测试内容 | 检测游戏能否正常重新开始 |
| 在游戏结束时按Q键 | 游戏正常退出并正常输出测试内容 | 检测游戏能否正常退出 |
| —— | 提示激光与真实激光都能正常出现，测试结果正常输出 | 检测激光是否能够正常生成 |
| 移动角色与真实激光重合 | 游戏结束，并且测试结果正常输出 | 检测碰撞部分是否成功 |
| 移动角色不断躲避激光 | 得分正常增加，测试结果正常输出 | 检测得分系统是否正常 |

### 测试结果

输入对应输出均正常，在应用程序输出栏目也可以看到测试信息的正常输出。说明程序本身运行正常。

1. **收获**
2. **对C++编程语法的进一步熟悉**

通过qt编程进一步熟悉了C++语法当中类与对象，指针等方面的编程语法。特别是熟悉了C++中浮点数与整型数对于实际精度的影响与如何规避强制转化实现程序的更高精度。

**2.单元测试对于编程十分重要**

在实际编程当中如果要更新一个功能可以添加单元测试，通过在应用程序输出栏目检查是否有测试输出就可以判定该功能是否在程序当中正常发挥作用。同时如果程序存在问题并且找不到的时候也可以添加单元测试，通过各部分在应用程序当中的输出就可以判断哪一部分正常运行哪一部分运行失败从而排查出问题。

### **3.定时器与信号槽的管理**

熟练掌握了 Qt 中定时器的使用方法，包括设置定时器的间隔、启动和停止定时器。理解了信号槽机制在事件驱动编程中的重要性，学会了如何正确连接和断开信号槽，避免内存泄漏和逻辑错误。学会了如何在定时器的回调函数中处理复杂的逻辑，如生成激光、更新倒计时、检测碰撞等。

1. **图形效果的实现**

了解了 Qt 中图形渐变的使用方法，通过设置渐变的颜色和位置，实现不同的视觉效果。掌握了如何为图形项添加发光效果和动画效果，如使用 QGraphicsDropShadowEffect 和 QPropertyAnimation。学会了如何通过调整图形项的属性，如透明度和颜色，实现动态的视觉效果。