中国传媒大学

信息与通信工程学院

结课作业报告

|  |  |
| --- | --- |
| 班级： | 数字媒体技术3班 |
| 姓名： | 刘胤吉 |
| 学号： | 202411023001 |

|  |  |
| --- | --- |
| 项目名称： | 基于QT开发的画图软件 |

项目类型：单人项目

2025年5月20日

## **基于 QT的画图软件设计开发实验报告 一、题目与内容**

**题目**：基于QT6的绘图软件设计与实现  
**内容**：本实验设计并实现了一个功能完善的绘图软件，采用C++面向对象编程思想，基于QT6.9.0框架开发。软件实现了多种绘图工具、历史记录管理、颜色选择、图像保存与加载等功能，所有UI组件均通过代码实现，不依赖外部资源文件。

## **二、 运行环境**

### **硬件环境** 处理器：Intel Core i5及以上 内存：8GB及以上 显示器：分辨率1920×1080及以上

### **软件环境** 操作系统：Windows 10/11 开发环境：Qt Creator 16.0.2 编译器：MinGW 64-bit Qt版本：Qt 6.9.0 构建系统：qmake 开发语言：C++

## **三、课题分析**

**基本要求：**

定义shape类，矩形类、椭圆类等。在QT开发环境下，开发具有画图功能的软件，能够绘制不同类型的几何图形。

功能要求：

1、设计菜单实现功能选择；

2、用鼠标实现几何图形的绘制；

3、保存绘制的图像功能：

4、打开已保存图像，并继续绘制。

5、拓展功能：

1）绘制复杂几何图形，如五角星、箭头等；

2）图元编组功能

3）撤销/重绘功能

4）其他

**功能结构分析：**

该画图软件主要由以下几个功能模块组成：

1. **历史记录管理模块**：负责记录的操作历史，并提供撤销和恢复功能。
2. **颜色选择模块**：提供颜色选择界面，可以选择画笔的颜色。
3. **画笔设置模块**：允许调节画笔的粗细。
4. **图形绘制模块**：实现各种图形的绘制功能，包括矩形、椭圆形、直线、箭头、五角星形、菱形、心形等。
5. **橡皮擦模块**：实现橡皮擦功能，擦除不需要的部分。
6. **文件管理模块**：负责绘画作品的保存和打开功能。
7. **图元编组移动模块**：实现图元的编组和移动功能。

**系统架构设计：**系统采用三层架构：  
**1.**表示层**：MainWindow类** 工具栏管理  
 状态栏显示  
 用户输入处理  
**2.**业务逻辑层**：PaintArea类** 绘图状态管理  
 命令执行与撤销  
 坐标系统转换  
**3.**数据层**：Shape类层次结构** 图形数据存储  
 绘制算法实现  
 边界计算

## 

## **四、主要功能模块，类图说明，程序运行流程，**

### **主要功能模块：**

1. ****Shape 类及其派生类**：**

**Shape 类**：作为抽象基类，定义了所有图形的基本属性和操作，如起始点、结束点、颜色、画笔宽度等，以及纯虚函数 draw ()、boundingRect ()、update () 和 clone ()。

**LineShape 类**：继承自 Shape 类，实现直线的绘制功能。

**RectangleShape 类**：继承自 Shape 类，实现矩形的绘制功能。

**EllipseShape 类**：继承自 Shape 类，实现椭圆形的绘制功能。

**ArrowShape 类**：继承自 Shape 类，实现箭头的绘制功能。

**StarShape 类**：继承自 Shape 类，实现五角星形的绘制功能。

**DiamondShape 类**：继承自 Shape 类，实现菱形的绘制功能。

**HeartShape 类**：继承自 Shape 类，实现心形的绘制功能。

**PathShape 类**：继承自 Shape 类，用于自由绘制和橡皮擦功能，通过保存一系列点来实现绘制。

1. ****PaintArea 类**：**

继承自 QWidget 类，是绘图区域的主要实现类。

负责处理鼠标事件（按下、移动、释放），根据用操作创建和绘制相应的图形。

实现图像的缩放和偏移处理，确保图像在不同大小的窗口中正确显示。

提供历史记录管理功能，包括撤销和恢复操作。

支持图元编组移动功能。

1. ****MainWindow 类**：**

继承自 QMainWindow 类，是主窗口类。

创建和管理工具栏和状态栏，提供用户交互界面。

处理用户的菜单和按钮点击事件，如颜色选择、画笔粗细调节、图形选择、文件保存和打开等。

### 

### **类图及编程思想说明： 类图：**

### 类图1 **编程思想说明： 1.**封装**：** 每个类都将其数据成员封装在类内部，通过公共成员函数提供访问和操作接口，如 Shape 类封装了图形的基本属性和操作。 ****PaintArea类的封装****： ****数据隐藏**：**将核心绘图数据（如image、originalImage、tempImage）声明为私有成员，外部只能通过公共接口访问 ****接口设计**：**提供公共方法控制绘图行为 ****状态管理****：封装撤销/重做栈的实现细节 **MainWindow类的封装 **UI控制隔离**：**所有UI组件创建和布局逻辑封装在私有方法中 ****业务逻辑封装**：**将文件操作、颜色选择等逻辑封装为独立方法 ****2.继承**：** LineShape、RectangleShape、EllipseShape 等类继承自 Shape 类，继承了 Shape 类的属性和方法，并根据自身特点进行了扩展和实现。 **Shape基类实现：** 定义图形通用接口和基本属性 **具体图形类的继承实现** **LineShape：**实现直线绘制逻辑 **PathShape：**特殊处理自由绘制和橡皮擦 **HeartShape：**实现较为复杂的贝塞尔曲线绘制

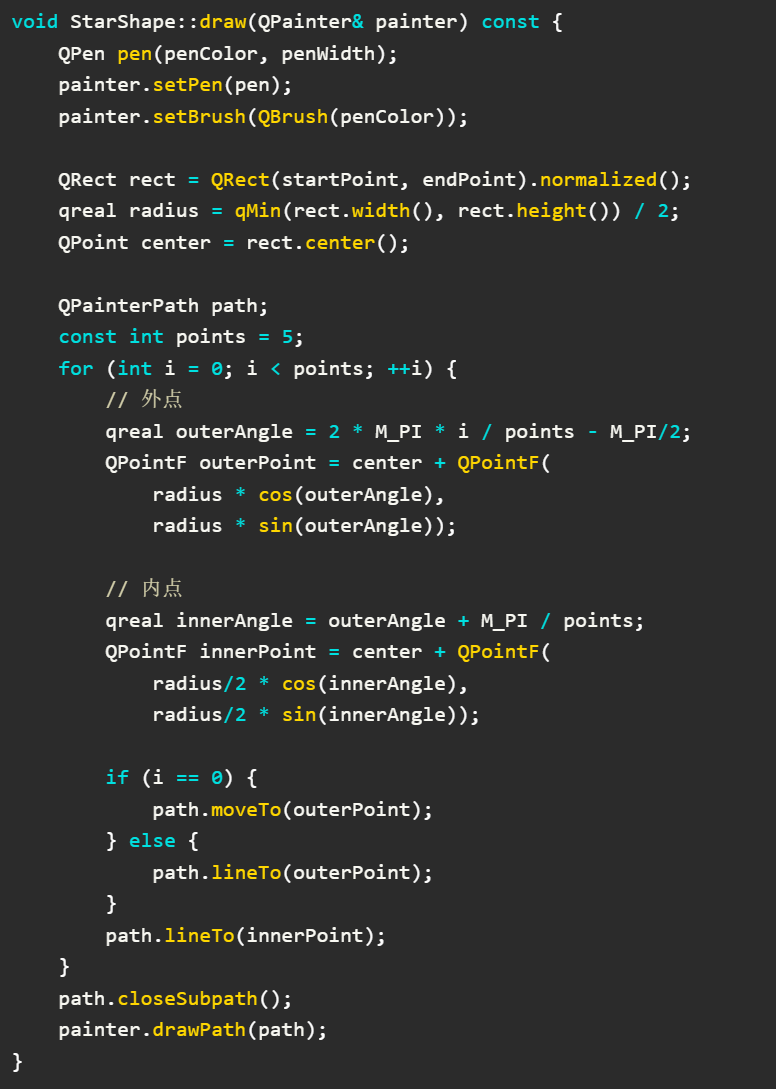
### ****3.多态：**** Shape 类中的 draw ()、boundingRect ()、update () 和 clone () 等函数为纯虚函数，子类通过重写这些函数实现了多态性。在 PaintArea 类中，可以通过基类指针调用不同子类的这些函数，实现不同图形的绘制和操作。

### ****运行时多态实现****： **虚函数表机制：**Shape类声明关键虚函数 **统一接口调用：**PaintArea通过基类指针操作具体图形 **对象克隆支持：**通过clone（）实现原型模式 **4.组合： MainWindow和PaintArea的组合 强生命周期管理：**MainWindow作为父对象拥有PaintArea **功能委托：**将绘图功能完全委托给PaintArea **PaintArea和Shape的组合 动态对象管理：**运行时创建和管理Shape派生类对象 **多态容器支持：**历史记录栈储存绘图操作 **程序运行流程：** 绘图流程：

#### 流程图1

## **五、重要功能代码实现细节举例**

**1.多态图形工厂实现**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**2. 历史记录管理功能，撤销系统实现**

**3.复杂图形绘画算法实现（以五角星为例）**

## **六、调试、测试记录及分析**

### **调试过程**

在开发过程中，主要使用 QT Creator 的调试工具进行调试。通过设置断点、查看变量值和调用栈等方式，定位和解决了以下问题：

1. **图形绘制问题**：在绘制某些图形时，发现图形的位置和大小不符合预期。通过调试发现，是由于坐标转换和边界计算错误导致的。经过修正后，图形绘制正常。
2. **历史记录管理问题**：在撤销和恢复操作时，发现图像显示异常。通过调试发现，是由于图像状态保存和恢复的逻辑错误导致的。经过修改后，历史记录管理功能正常。
3. **鼠标事件处理问题**：在处理鼠标事件时，发现图形绘制不连续或出现闪烁现象。通过调试发现，是由于临时图像更新和重绘的时机不正确导致的。经过调整后，鼠标事件处理正常。

### **测试过程**

对软件进行了以下方面的测试：

1. **功能测试**：对软件的各项功能进行了全面测试，包括历史记录管理、颜色选择、画笔设置、图形绘制、橡皮擦、文件保存和打开、图元编组移动等功能。测试结果表明，各项功能均能正常工作。
2. **兼容性测试**：在不同的 Windows 操作系统版本和屏幕分辨率下进行了测试，软件均能正常运行，未出现明显的兼容性问题。
3. **压力测试**：进行了长时间的连续绘图和频繁的撤销、恢复操作，软件的性能表现良好，未出现卡顿或崩溃现象。

### **代码管理与显示优化 利用Github对软件代码进行储存和管理，方便保存记录和调试 f782688727b360622d7bb4b9135ed00ee39b231bf01dbe28c5b59fcc2a8dc4 对界面ui进行优化调整，提升操作体验 1.1版本（基础控件） 633de9ddd3d7802ccf90b5aa9d2e93e**

### **1.2版本（优化视觉效果） 80eee5c4b2f6244ba678f33ef5aaf06 1.3版本（增添图标，增加操作指引）**

****

## **七、软件运行测试，BUG 记录和改进**

### **BUG 记录**

1. 在进行图元编组移动时，如果选择的区域超出了图像边界，会导致程序崩溃。
2. 在撤销和恢复操作时，有时会出现图像显示不完整的情况。
3. 在使用橡皮擦功能时，擦除效果不够平滑，会出现锯齿状边缘。

### **运行记录及分析**

在实际运行过程中，软件的界面响应速度较快，图形绘制流畅。用户可以方便地使用各种绘图工具和功能，满足了基本的绘图需求。然而，在绘制复杂图形或进行大量操作时，软件的性能会有所下降，这可能是由于图形绘制和图像保存的算法复杂度较高导致的。

### **改进措施**

1. **性能优化**：对图形绘制和图像保存的算法进行优化，减少不必要的计算和内存开销，提高软件的性能。
2. **用户体验改进**：增加更多的绘图工具和功能，如填充颜色、渐变效果、文字输入等，提高软件的实用性和趣味性。
3. **稳定性提升**：修复已知的 BUG，加强错误处理和异常处理机制，提高软件的稳定性和可靠性。
4. **界面美化**：对软件的界面进行美化，采用更加美观和易用的设计风格，提高用户的视觉体验。

## **八、总结**

### **优点**

1. **面向对象编程思想的应用**：通过使用类和继承、多态等面向对象编程的特性，将不同的功能封装在不同的类中，提高了代码的可维护性和可扩展性。
2. **功能丰富**：软件具备多种绘图功能，如历史记录管理、颜色选择、画笔设置、多种图形绘制、橡皮擦、文件保存和打开、图元编组移动等，满足了用户的基本绘图需求。
3. **跨平台性**：基于 QT 框架开发，具有良好的跨平台性，可以在不同的操作系统上运行。

### **不足**

1. **性能问题**：在同时处理复杂图形和大量操作时，软件的性能会有所下降。
2. **用户体验**：软件的界面设计和操作方式还有待改进，用户体验不够友好。
3. **功能完整性**：软件的功能还不够完整，如缺乏填充颜色、渐变效果、文字输入等高级绘图功能。

### **收获及体会**

通过本次实验，我对 C++ 语言和 QT 框架有了更深入的了解和掌握。在开发过程中，我深刻体会到了面向对象编程思想的重要性，它可以使代码更加清晰、易于维护和扩展。同时，我也学会了如何使用 QT Creator 进行软件开发，包括界面设计、事件处理、文件操作等方面的知识。在调试和测试过程中，我积累了丰富的经验，提高了自己解决问题的能力。此外，通过不断地优化和改进软件，我也体会到了软件开发是一个不断完善的过程，需要持续地学习和实践。