北京邮电大学软件学院

实验报告

（注意：此处填写实验指导手册中的实验名称）

**提交时，将本注释框删除**

课程名称： C语言程序设计课程设计

项目名称：

项目完成人：

姓名： 姜洪烨 学号： 2019211915

指导教师： 杨 谈

日 期： 2019 年 12 月 日

# 实验目的

利用EGE (Easy Graphics Engine) 和C/C++来实现图形界面画板的功能。

# 实验内容

（简要说明本实验的内容）

# 实验环境

操作系统：Windows 10 Enterprise LTSC

IDE：Visual Studio 2019 Community

# 附录

（附上实验文档，如：问题分析、设计方案、算法、设计图、程序、仿真结果、运行结果、调试心得等，具体内容根据实验要求来定。源代码请附在这里。源代码排版请特别注意，用5 号字体，行间距为单倍行距。注意节省空间，不要浪费纸张。）

## 问题分析

（说明你对这个问题的理解，包括，这个程序要解决什么问题、功能、性能、健壮性等）

### 目标

（说明这个程序要解决什么问题，达到的目标）

### 功能

（说明这个程序的各种功能，参见下例4.1.2.1和4.1.2.2。在撰写正文时，请将下面的例子删除）

#### 功能一：撤回

用户可以在图形绘制界面点击撤回按钮实现无限制的撤回操作。

#### 功能二：前景和填充色自定义

用户可以从调色盘中精确点击需要的颜色或者从预设中选择。

#### 功能三：自定义存档保存与打开

用户可以自定义保存位置并保存当前绘图存档，并且在下次可选择要加载的存档，打开后仍可以进行撤回等编辑存档的操作。

#### 用鼠标绘制圆、线、矩形

用户只需在屏幕上点击即可选定起点/终点、圆心/半径、矩形的左上角/右下角，通过极为快速简单的操作即可绘制图形，并且用户在移动鼠标时会有即将绘制的图形的实时预览，不用担心点击的位置不准确。

#### 用鼠标绘制多边形

用户只需在屏幕上点击各个顶点，最后一个顶点靠近第一个顶点时即可自动吸附并且闭合多边形，直观地完成绘图

#### 输入坐标绘制多边形

通过完整的输入格式检查和提醒，用户可以轻易地通过输入坐标来绘制多边形。

### 性能

以下性能测试基于的硬件平台：

CPU: Intel Core i5-7200U at 2.50GHz with TurboBoost Up to 3.10GHz

Memory: 32GB DDR4 at 2133MHz Dual Channel Interlaced

Graphics: Intel HD Graphics 620

#### 画面更新性能

在鼠标移动或者画图形后的画面更新帧率至少在800 FPS (Frame Per Second)，即更新画面的时间至多1.25 ms。无肉眼可以感知的延迟。

### 健壮性

#### 写入或读取存档时的异常处理

##### 第一层：考虑到在程序中调用了Windows API来让用户自由读取存档，因此有必要在用户选择错误的文件时报错。在本项目中通过在保存文件时添加校验位来标记该文件是本程序创建的。在读取时通过对检验位的比较来辨别用户选择的文件正确与否。在用户选择错误的文件时会弹出警告窗口来使用户重试。

##### 第二层：在写入或读取存档时会对成功写入或读取的数据块的数量与应该成功的数据块数量进行比较，在写入或读取时有数据块出错时弹出警告提醒用户重新操作。

#### 在用户键入多边形边数或坐标时的异常处理

##### 对于边数的异常处理

###### 检查输入字符串是否全为数字，并提示错误点

作为边数，自然都必须为数字，程序将检查输入的字符串是否都为数字，再使用atoi()函数转换，否则警告"边数只能包含数字"

###### 检查边数在几何意义上的合理性，并提示错误点

从几何意义上说，多边形的边数必须大于等于3，若用户输入的边数小于3，则警告"不能构成多边形"；若多边形的边数大于24，即超出定义的数组能够存储的坐标范围，为防止溢出，将警告“坐标个数过多”。

##### 对于坐标的异常处理

在按照 ”,” 分割输入的字符串后将检查：

###### 检查分割后的字符串是否全为数字，并提示错误点

作为坐标，自然都必须为数字，程序将检查输入的字符串是否都为数字，再使用atoi()函数转换，否则警告"坐标只能包含数字"

###### 检查坐标个数，并提示错误点

一个边数为sides的多边形只能有sides个坐标，若读取的坐标数量过多，则警告"坐标个数过多"；若读取的坐标数量过少，则警告"坐标个数过少"

###### 检查坐标范围，并提示错误点

若输入的坐标超出屏幕的显示范围，则警告"坐标范围无效"

##### 对于用鼠标画多边形时坐标个数溢出的处理

由于数组的限制，程序能够存储的坐标个数有限，在用户画图时，坐标的个数将被实时检查，在即将溢出时会提示用户将多边形封闭停止作画。

##### 对于图形数量过多溢出的处理

介于对内存空间节省和正常情况下会存储的图形数量的考虑，程序设计时保存图形的最大数量为512个，在用户画的图形数量即将超出这个限制时将提示用户及时停止绘画并保存图形。

## 设计方案

整个项目分为9个模块，对应9个C++源代码文件和9个头文件，每个模块有2-5个函数，共计25个函数。

（在此处，用文字描述整体的设计，包括，分成多少个模块，多少个文件，多少个函数，每个模块、函数、文件的大致功能； 此外，还需要包含一个程序整体的流程图，反映各个模块的执行次序和关系；但是避免绘制细节的流程）

### 模块、文件、函数的功能

文件功能：

color\_selector.cpp：包含颜色选取功能，颜色转换算法，打印颜色选取菜单。

color\_selector.h：对应的头文件。

coord\_draw.cpp：包含用坐标来进行绘画的功能，即用坐标绘制多边形。

coord\_draw.h：对应的头文件。

draw.cpp：包含最主要的根据数据绘制图形的功能，打印绘图菜单，随机颜色算法。

draw.h：对应的头文件。

ege\_based\_painter.cpp：包含main() 函数，全局变量。

global.h：包含宏定义，结构体定义等。

menu.cpp：包含功能菜单选取，打印菜单的功能。

menu.h：对应的头文件。

mouse\_draw.cpp：包含用鼠标来进行绘画的功能，即用鼠标来绘制圆、线、矩形、多边形。

mouse\_draw.h：对应的头文件。

read\_file.cpp：包含加载存档、将存档写入内存的功能。

read\_file.h：对应的头文件。

save\_file.cpp：包含将当前绘图状态保存至存档的功能。

save\_file.h：对应的头文件。

UI.cpp：包含对于基本用户界面的初始化、显示状态栏（显示系统状态、图形数量分类统计、鼠标坐标）、获取鼠标所在的菜单、清除图形等的功能。

UI.h：对应的头文件。

### 运行简化主体流程图

## 重要算法

（在此处，用流程图描绘重要的算法，例如排序、查找等。一般的程序代码不必在此说明）

### 算法一：对于特定图形类型的数量统计算法

static WORD tmp\_totalShapes;

static WORD nLines;

static WORD nCircles;

static WORD nRectangles;

static WORD nPolygons;

// count the number of each shape

if (tmp\_totalShapes != g\_nTotalShapes)

{

fileEdited = true;

nLines = 0;

nCircles = 0;

nRectangles = 0;

nPolygons = 0;

for (int i = 0; i < g\_nTotalShapes; i++)

{

switch (shapeData[i].shapeType)

{

case shape\_line:

nLines++;

break;

case shape\_circle:

nCircles++;

break;

case shape\_rectangle:

nRectangles++;

break;

case shape\_polygon:

nPolygons++;

break;

default:

break;

}

}

}

### 算法二：绘制颜色选择面板的算法

void PrintColorPanel(WORD x\_offset, WORD y\_offset)

{

unsigned int color\_x, color\_y, colorPanelValue;

for (color\_y = 0; color\_y < 360; ++color\_y)

{

for (color\_x = 0; color\_x < 256; ++color\_x)

{

colorPanelValue = hsl2rgb((float)color\_y, 1.0f, 0.5f);

putpixel((int)(0.9 \* color\_x + x\_offset), (int)(0.9 \* color\_y + y\_offset), alphacol(colorPanelValue, 0x000000, color\_x));

}

}

}

color\_t alphasingle(color\_t a, color\_t b, color\_t alpha)

{

return (a \* (0xFF - alpha) + b \* alpha) >> 8;

}

color\_t alphacol(color\_t d, color\_t s, color\_t alpha)

{

return (alphasingle(d & 0xFF00FF, s & 0xFF00FF, alpha) & 0xFF00FF)

| (alphasingle(d & 0xFF00, s & 0xFF00, alpha) >> 8 << 8);

}

## 实验结果

（说明实验完成情况，有需要文字回答的题目，以及实验结果截图，请写在这里）

（在此处，详细描述运行的结果。包括：不同输入对应的不同结果，异常情况的执行结果）

## 调试心得

（总结你在调试程序时的收获，对于某一类或者某几类警告、错误的处理方法。这些心得，能够对学习C语言程序设计的新手具有一些指导作用）