北京邮电大学软件学院

实验报告

课程名称： C语言程序设计课程设计

项目名称： 画板程序

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 24 日

# 实验目的

在编写画板程序的过程中综合运用并提升C语言编程技能与规范，为以后的学习做好铺垫。

# 实验内容

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

# 实验环境

OS：Windows 10 Enterprise LTSC version 1809

IDE：Visual Studio 2019 Community 16.3.9

# 附录

## 问题分析

画板即一个可以将用户输入的数据转化为屏幕上的图像的应用程序。本项目中使用C语言完成。它需要具有以下几个功能：1、拥有自然的数据输入方式。2、拥有存档功能。3、拥有人性化的用户界面。4、拥有多种图形的作画功能。5、可以自定义作图参数。

### 目标

使用C语言和EGE库完成画板程序，即一个可以将用户输入的数据转化为屏幕上的图像的应用程序。

### 功能

#### 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。

#### 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。

#### 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，并且可以自由选择要加载的存档，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，非常直观，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

通过较为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：

Processor: 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1000FPS (Frame Per Second)，即更新一帧画面的时间至多1ms。无可感知的延迟。

### 健壮性

#### 写入或读取存档时的异常处理

##### 在读取到非存档文件时的处理

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。

##### 读取或写入失败时的异常处理

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

##### 在读取或写入文件后关闭文件失败的处理

若关闭文件失败，则提示用户重新操作

#### 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

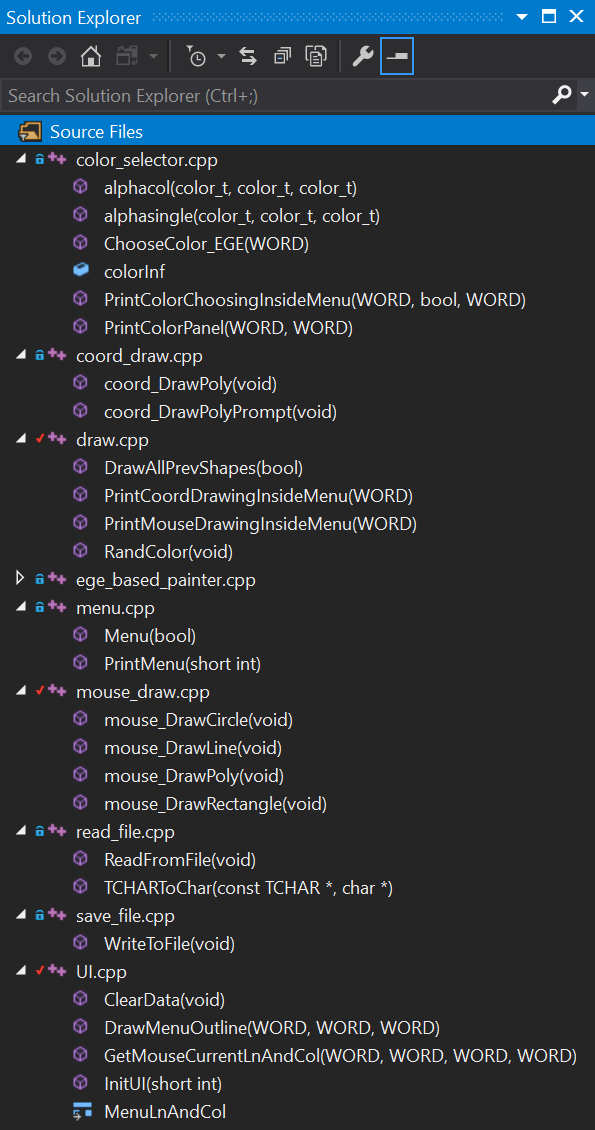
整个项目分为9个模块，对应9个C++源代码文件和9个头文件，每个模块有2-5个函数，共计25个函数。

### 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 |

#### 函数功能



函数名称即该函数的用途，无需赘述。

### 运行简化主体流程图



## 重要算法

### 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：鼠标移动时菜单高亮与鼠标画图的算法

void mouse\_DrawPoly(void)

{

const short int TOTAL\_LN = 4; // total items in the menu bar

bool isInProgress = false; // To determine whether the mouse click is the first step or the second step

WORD polyCoords[50];

WORD sides = 0;

printf("已进入鼠标画多边形模式\n");

printf("操作指南：\n");

printf("用鼠标点选顶点，最后一个点靠近起始点来结束\n");

DrawAllPrevShapes(true);

DrawMenuOutline(1, TOTAL\_LN, 1);

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

mouse\_msg msg;

for (; is\_run(); delay\_fps(REFRESH\_RATE)) // Using "for" statement to draw multiple circles at a time and refresh the screen

{

msg = getmouse();

switch (msg.msg)

{

case mouse\_msg\_down:

switch (GetMouseCurrentLnAndCol(1, TOTAL\_LN, 1, 1).ln)

{

case 1:

return;

break;

case 2:

if (g\_nTotalShapes > 0)

{

if (!isInProgress)

{

g\_nTotalShapes--;

}

}

// refresh the windows with menu contents

cleardevice();

InitUI(0);

DrawMenuOutline(1, TOTAL\_LN, 1);

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

setcolor(0x50AA50);

xyprintf(678, 582, "当前坐标: (%03d, %03d)", msg.x, msg.y);

DrawAllPrevShapes(true);

goto move;

break;

case 3:

ChooseColor\_EGE(0);

cleardevice();

InitUI(0);

DrawAllPrevShapes(true);

goto move;

break;

case 4:

ChooseColor\_EGE(1);

cleardevice();

InitUI(0);

DrawAllPrevShapes(true);

goto move;

break;

default:

break;

}

/\* if the mouse click indicates the first dot, store the position data of the center of the circle \*/

if (!isInProgress)

{

polyCoords[0] = msg.x;

polyCoords[1] = msg.y;

sides = 0;

printf(" 您已选中点 (%d, %d)\n", polyCoords[0], polyCoords[1]);

isInProgress = true;

break;

}

/\* if the mouse click indicates the first dot, store the position data of other dots on the circle \*/

if (isInProgress)

{

sides++;

polyCoords[2 \* sides] = msg.x;

polyCoords[2 \* sides + 1] = msg.y;

printf(" 您已选中点 (%d, %d)\n", polyCoords[2 \* sides], polyCoords[2 \* sides + 1]);

if (sqrt(pow(polyCoords[0] - polyCoords[2 \* sides], 2) +

pow(polyCoords[1] - polyCoords[2 \* sides + 1], 2)) <= 8

&& sides >= 3)

{

polyCoords[2 \* sides] = polyCoords[0];

polyCoords[2 \* sides + 1] = polyCoords[1];

g\_nTotalShapes++;

shapeData[g\_nTotalShapes - 1].shapeType = shape\_polygon;

shapeData[g\_nTotalShapes - 1].extraData[0] = sides; // record the sides

// prepare the coordinate data for storage

for (int j = 0; j < shapeData[g\_nTotalShapes - 1].extraData[0]; j++)

{

shapeData[g\_nTotalShapes - 1].coords[j].x = polyCoords[2 \* j];

shapeData[g\_nTotalShapes - 1].coords[j].y = polyCoords[2 \* j + 1];

}

if (!g\_isUserSetColor)

{

shapeData[g\_nTotalShapes - 1].foregroundColor = RandColor();

}

else

{

shapeData[g\_nTotalShapes - 1].foregroundColor = g\_customColor;

}

if (!g\_isUserSetFillColor)

{

shapeData[g\_nTotalShapes - 1].isFill = false;

}

else

{

shapeData[g\_nTotalShapes - 1].isFill = true;

if (g\_isFillColorRandom)

{

shapeData[g\_nTotalShapes - 1].fillColor = RandColor();

}

else

{

shapeData[g\_nTotalShapes - 1].fillColor = g\_customFillColor;

}

}

DrawAllPrevShapes(true);

isInProgress = false;

printf(" 已完成%d边形的绘图\n", sides);

goto move;

}

if (sides >= 23)

{

MessageBox(NULL,

TEXT("边数过多，请立即将正在绘画的多边形封口"),

TEXT("即将溢出！"),

MB\_OK | MB\_SYSTEMMODAL | MB\_ICONEXCLAMATION);

}

break;

}

break; // not needed

case mouse\_msg\_move:

move:

if (!isInProgress)

{

InitUI(0);

//DrawMenuOutline(1, TOTAL\_LN, 1);

//setcolor(0x000000);

//PrintMouseDrawingInsideMenu(0);

setcolor(0x50AA50);

xyprintf(678, 582, "当前坐标: (%03d, %03d)", msg.x, msg.y);

switch (GetMouseCurrentLnAndCol(1, TOTAL\_LN, 1, 1).ln)

{

case 0:

DrawMenuOutline(1, TOTAL\_LN, 1);

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

break;

case 1:

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

setcolor(0x5050AA);

PrintMouseDrawingInsideMenu(1);

break;

case 2:

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

setcolor(0x5050AA);

PrintMouseDrawingInsideMenu(2);

break;

case 3:

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

setcolor(0x5050AA);

PrintMouseDrawingInsideMenu(3);

break;

case 4:

setcolor(0x000000);

PrintMouseDrawingInsideMenu(0);

setcolor(0x5050AA);

PrintMouseDrawingInsideMenu(4);

break;

default:

break;

}

}

if (isInProgress)

{

cleardevice();

delay\_fps(10000);

InitUI(1);

setcolor(0x50AA50);

xyprintf(678, 582, "当前坐标: (%03d, %03d)", msg.x, msg.y);

DrawAllPrevShapes(true);

setcolor(0x909090);

if (sides >= 1)

{

for (int i = 0; i < sides \* 2; i += 2)

{

line(polyCoords[i], polyCoords[i + 1], polyCoords[i + 2], polyCoords[i + 3]);

}

}

if (sqrt(pow((double)polyCoords[0] - (double)msg.x, 2) +

pow((double)polyCoords[1] - (double)msg.y, 2)) <= 8

&& sides >= 2)

{

line(polyCoords[sides \* 2], polyCoords[sides \* 2 + 1], polyCoords[0], polyCoords[1]);

}

else

{

line(polyCoords[sides \* 2], polyCoords[sides \* 2 + 1], msg.x, msg.y);

}

}

break;

default:

break;

}

}

}

### 算法三：对于鼠标所在菜单的采集算法

struct MenuLnAndCol GetMouseCurrentLnAndCol(

WORD lnStart,

WORD lnEnd,

WORD colNeeded,

WORD colTotal)

{

int x, y;

struct MenuLnAndCol coord;

coord.ln = 0;

coord.col = 0;

mousepos(&x, &y);

/\*

at first, i used the getmouse() function

but a i encountered a issue where the function keeps waiting mouse input when there is no mouse input

thus causing possible delays

later i switched to the mousepos() function

finally!!!!

there is no delay

\*/

if ((x >= 5 + (colNeeded -1) \* (MENU\_LENGTH / colTotal))

&& (y >= 5 + (lnStart - 1) \* MENU\_HIGHT)

&& (x <= 5 + (colNeeded) \* (MENU\_LENGTH / colTotal))

&& (y <= 5 + lnEnd \* MENU\_HIGHT))

{

coord.ln = (y - 5) / MENU\_HIGHT + 1;

coord.col = (x - 5) / (MENU\_LENGTH / colTotal) + 1;

}

return coord;

}

### 算法四：将读取的输入数组与(x,y)坐标数组互转的算法

(x.y)坐标数组转多边形绘图使用的数组

for (int j = 0; j < \*((shapeData + i)->extraData); j++)

{

\*(coordData + 2 \* j) = ((shapeData + i)->coords + j)->x;

\*(coordData + 2 \* j + 1) = ((shapeData + i)->coords + j)->y;

}

\*(coordData + 2 \* (\*((shapeData + i)->extraData))) = ((shapeData + i)->coords)->x;

\*(coordData + 2 \* (\*((shapeData + i)->extraData)) + 1) = ((shapeData + i)->coords)->y;

多边形绘图所用的数组转(x,y)坐标数组

for (int j = 0; j < shapeData[g\_nTotalShapes - 1].extraData[0]; j++)

{

shapeData[g\_nTotalShapes - 1].coords[j].x = polyCoords[2 \* j];

shapeData[g\_nTotalShapes - 1].coords[j].y = polyCoords[2 \* j + 1];

}

### 算法五：随机颜色算法

int RandColor(void)

{

int R = 0, G = 0, B = 0;

int hexColorValue; // the final hexadecimal output

do

{

randomize();

R = random(255);// randInt(0, 255);

randomize();

G = random(255);// randInt(0, 255);

randomize();

B = random(255);// randInt(0, 255);

}

while ((R >= 180) || (G >= 180) || (B >= 180)); // condition added to avoid the colors that are too dark. readibility improved.

hexColorValue = (R << 16) + (G << 8) + B; // convert separate RGB channels into a single value that represents colors

// printf("%x", hexValue); // display the output value of the function for debug purposes

return hexColorValue;

}

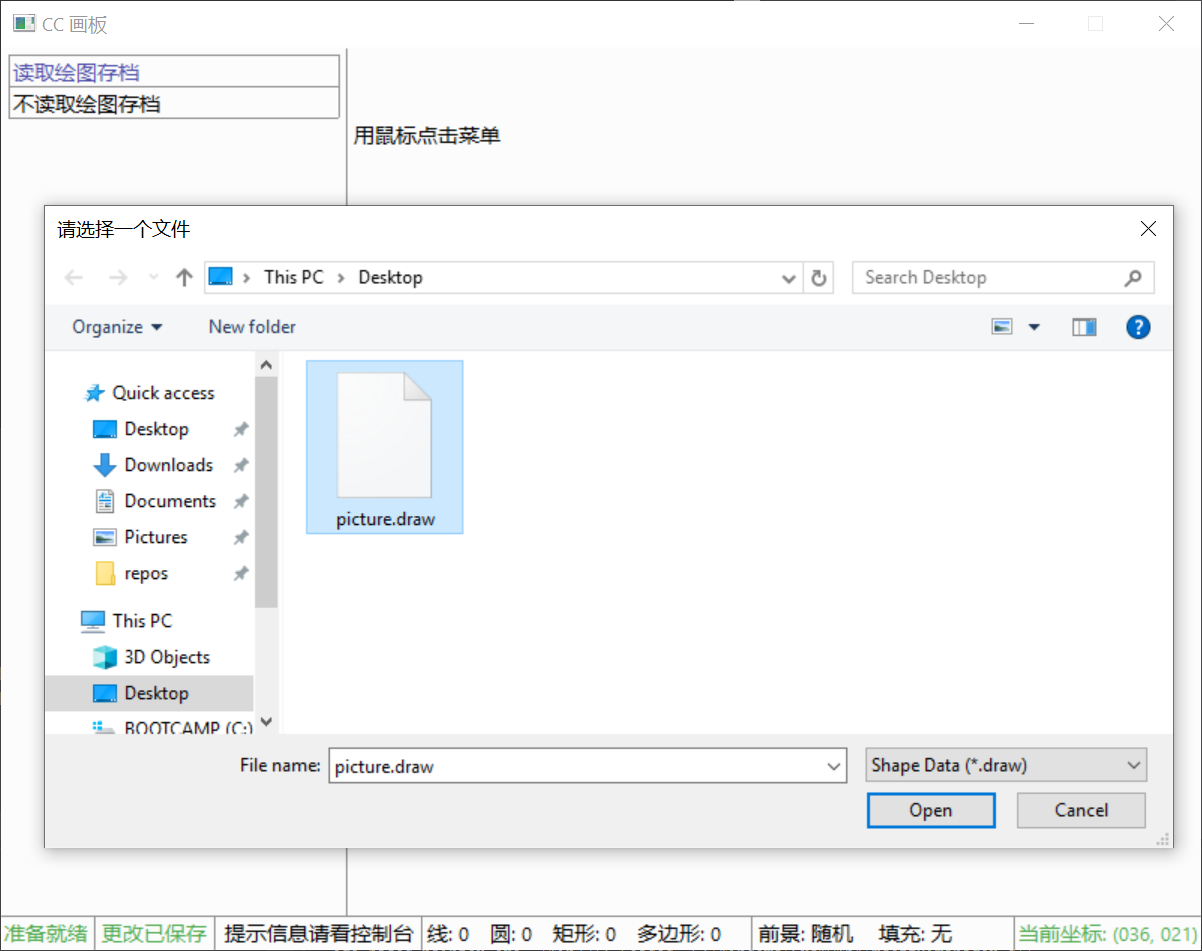
## 实验结果

### 基本功能

#### 读取文件菜单



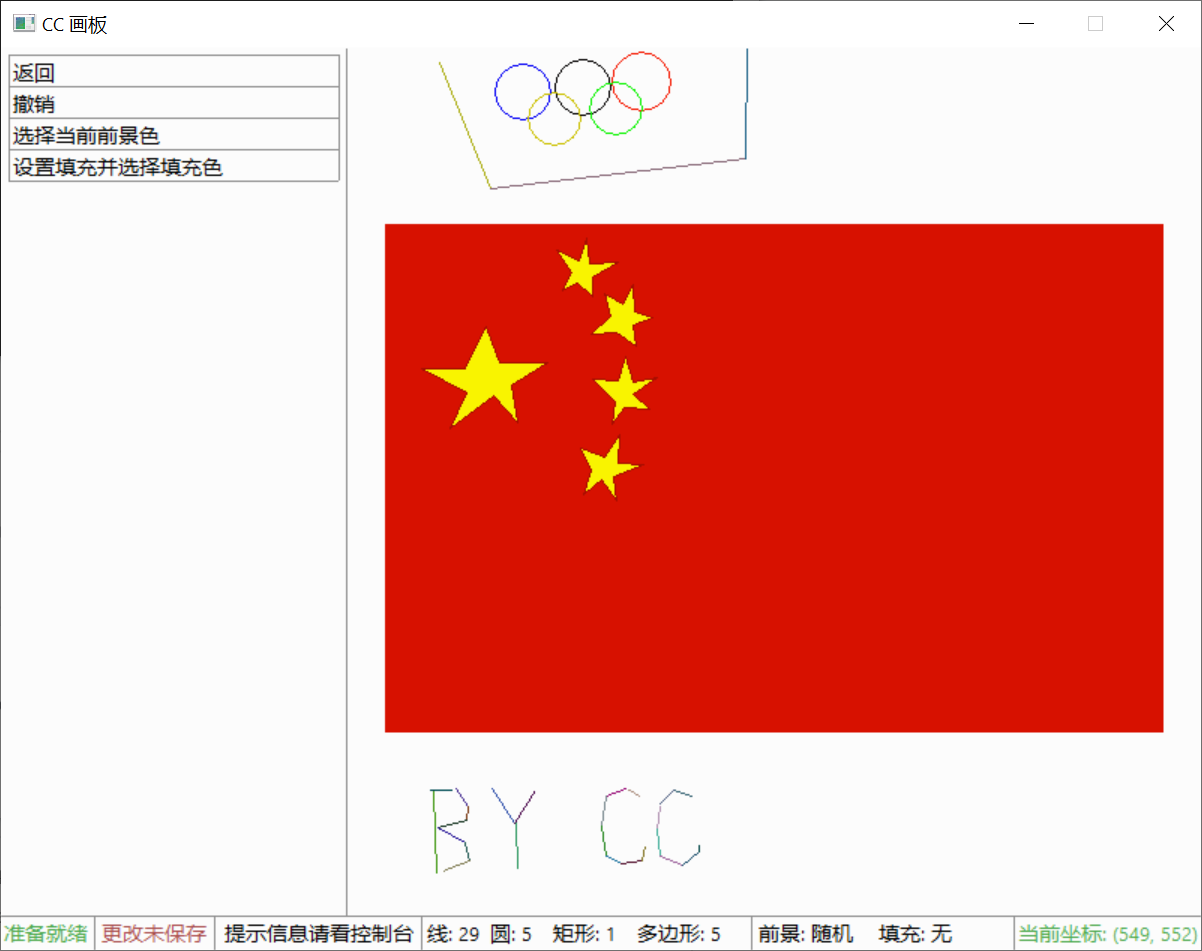
#### 读取存档

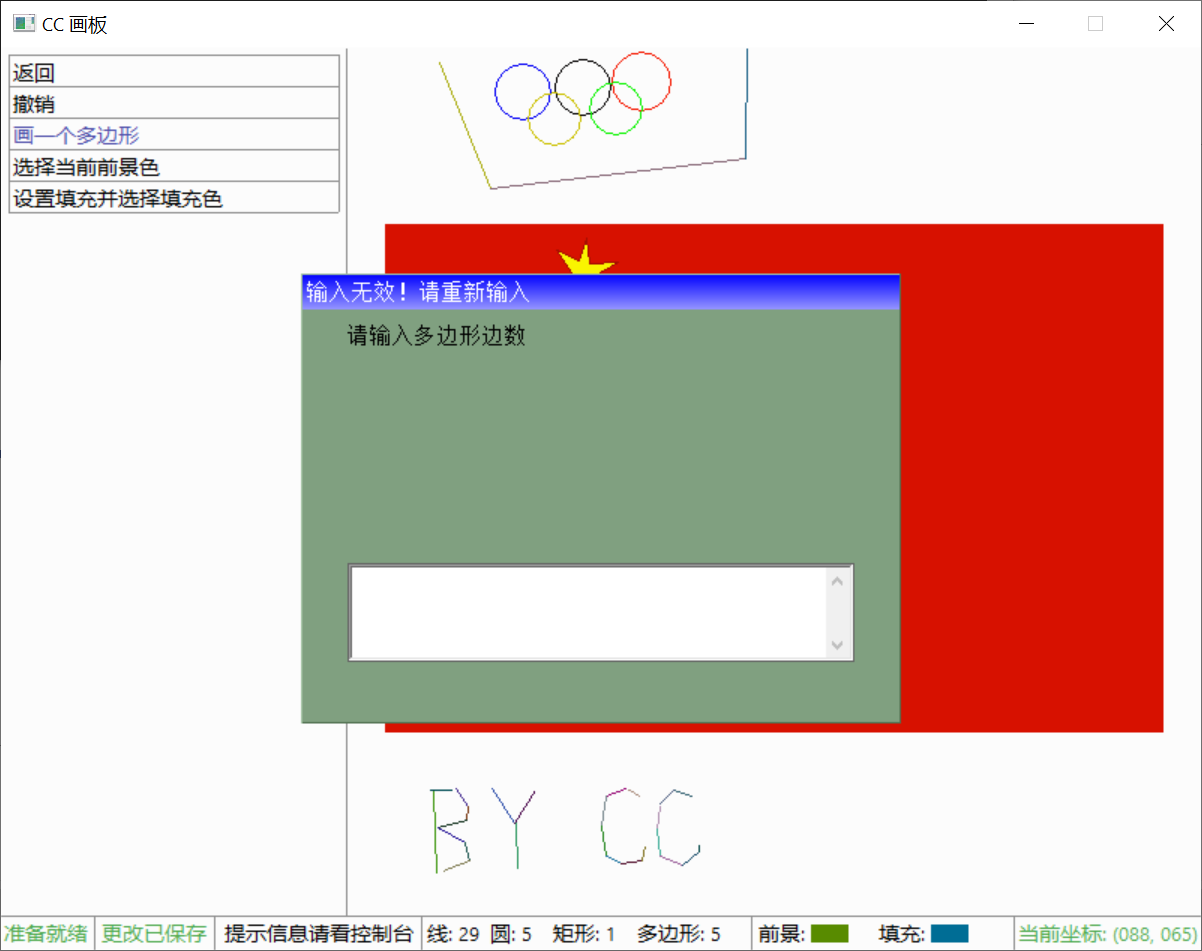


#### 加载存档后菜单

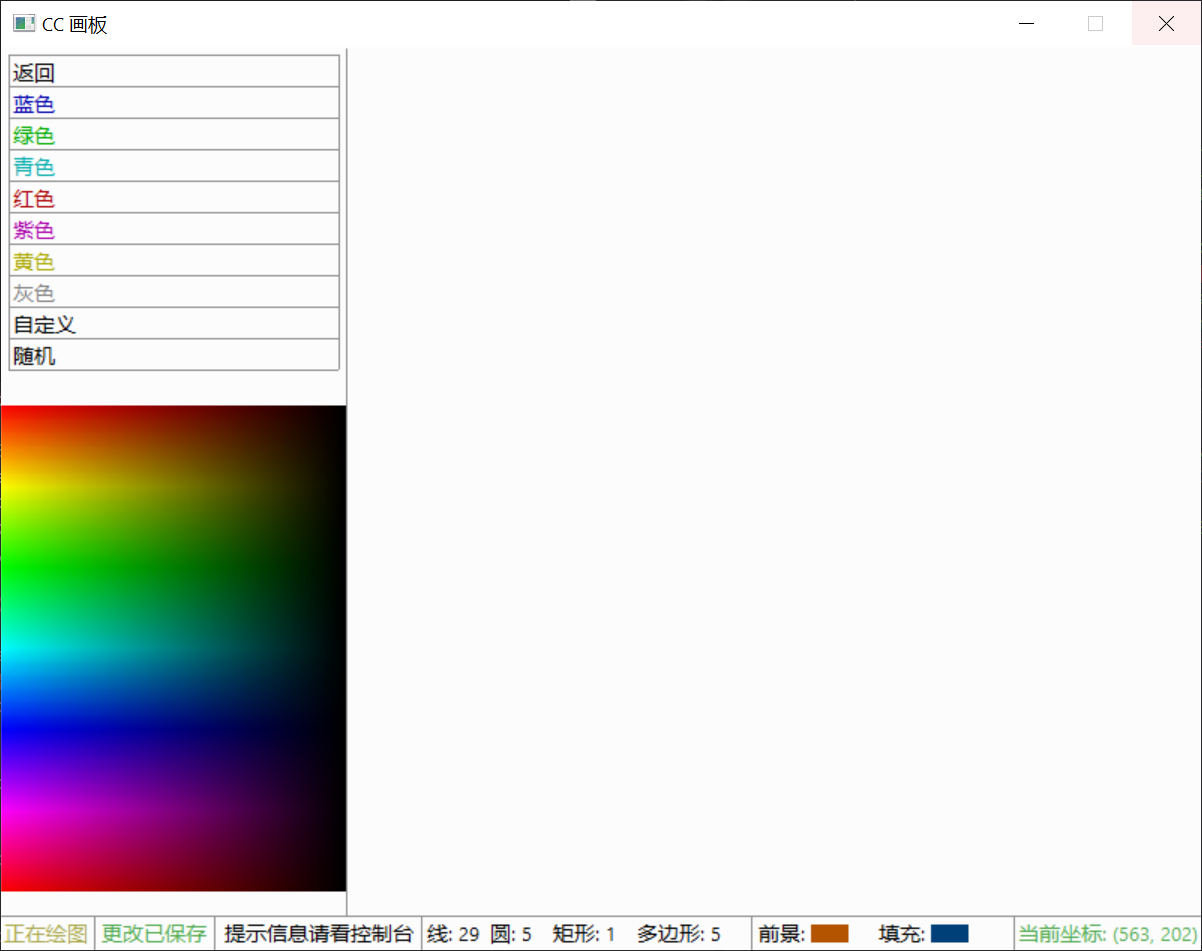


#### 绘图界面与菜单

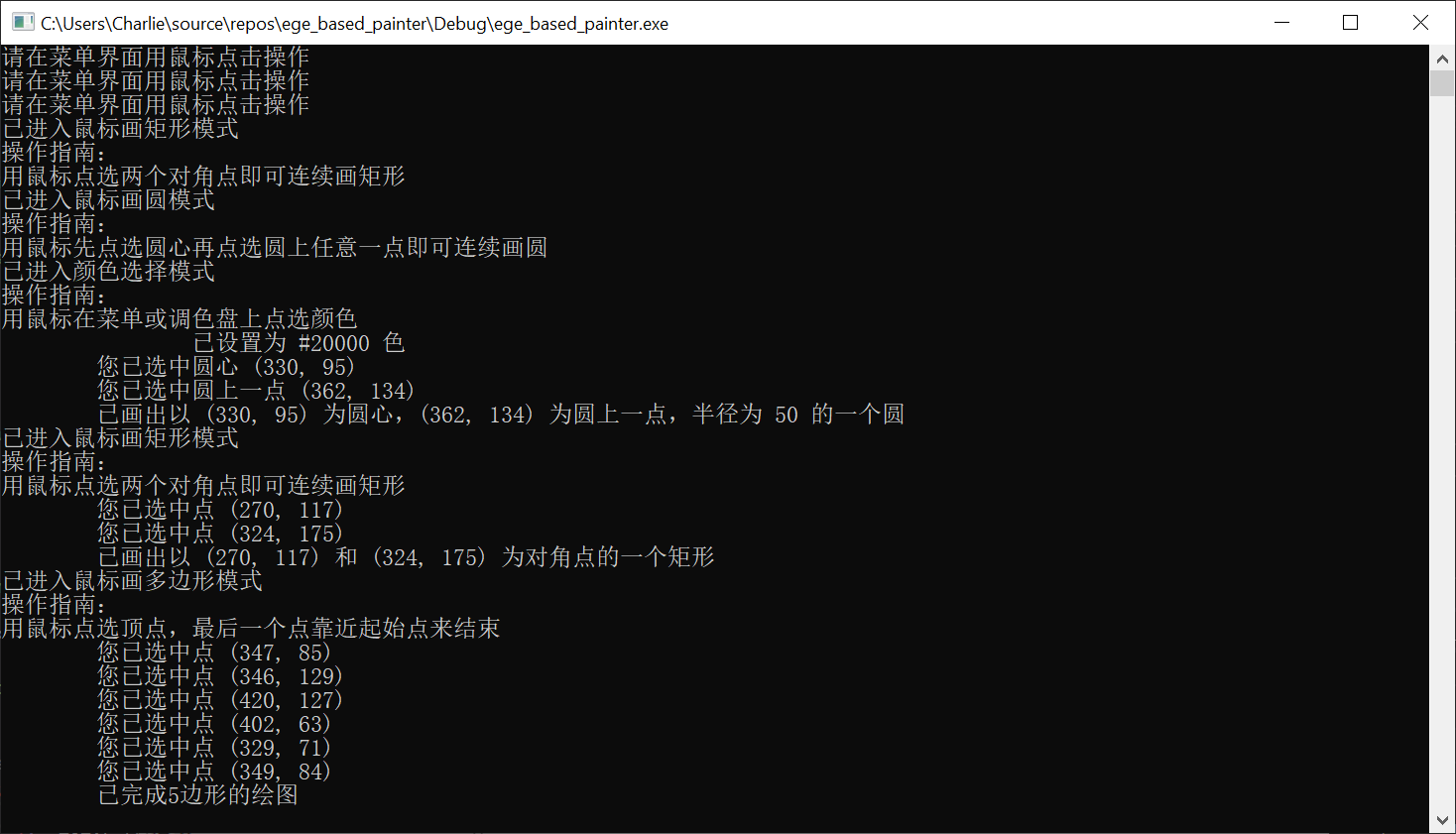




#### 颜色选择界面



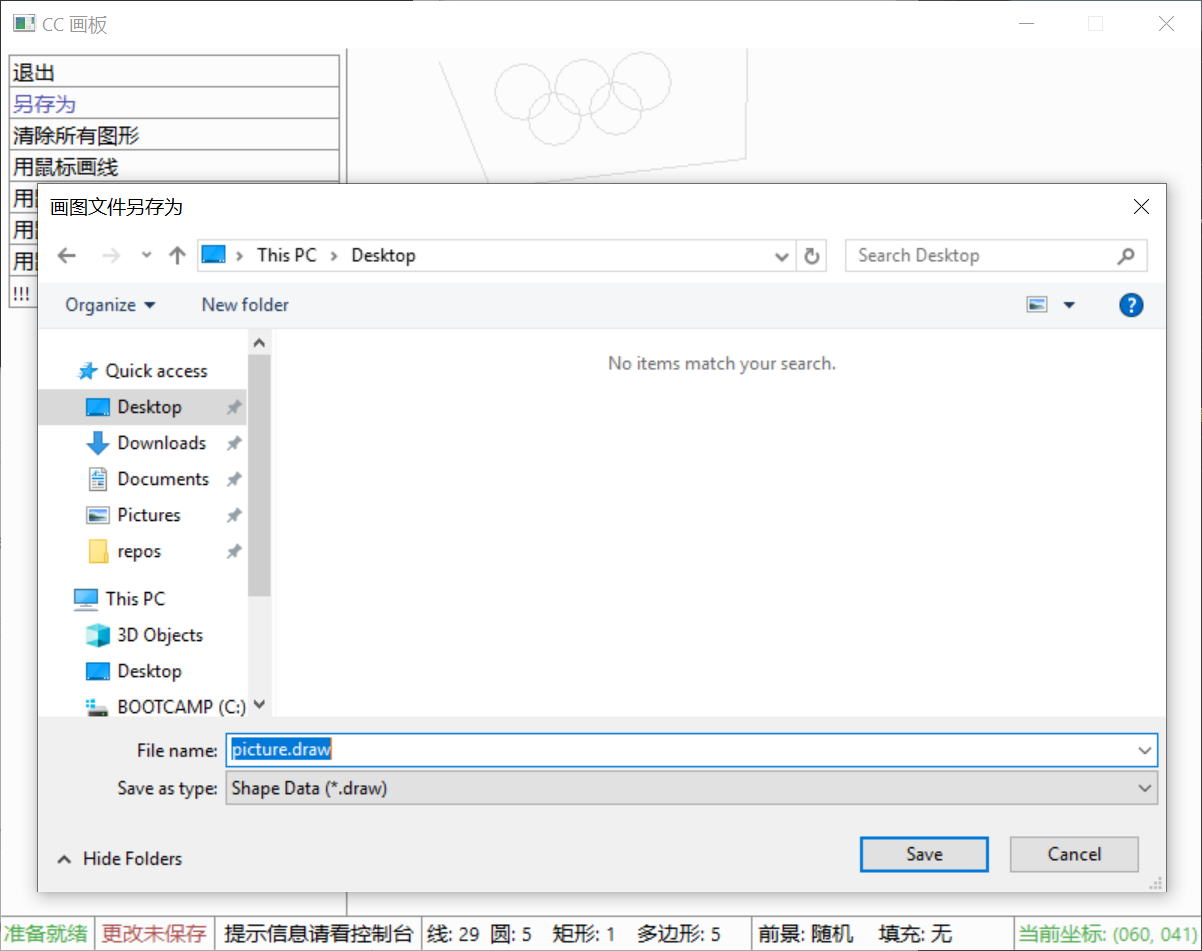
#### 绘图提示



#### 状态栏

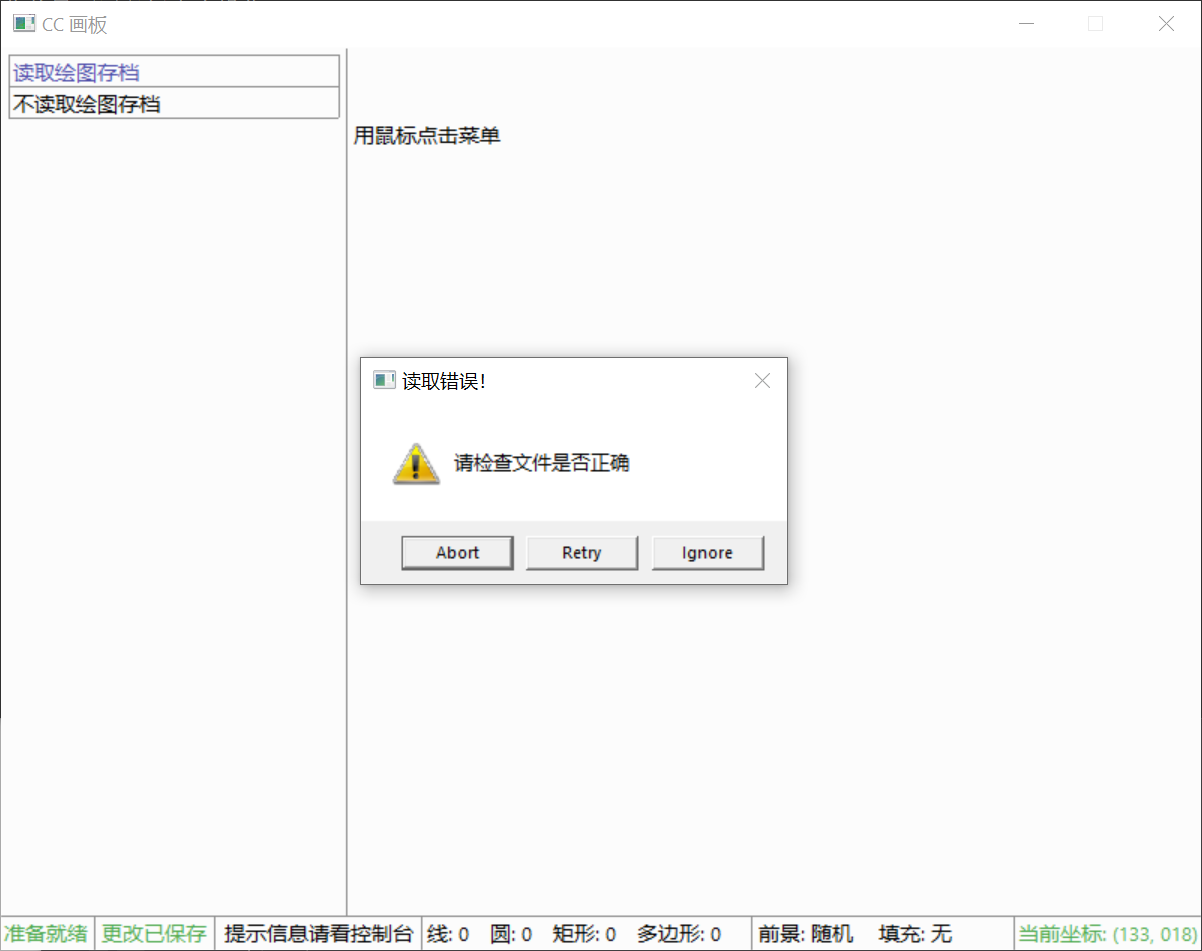


#### 保存存档

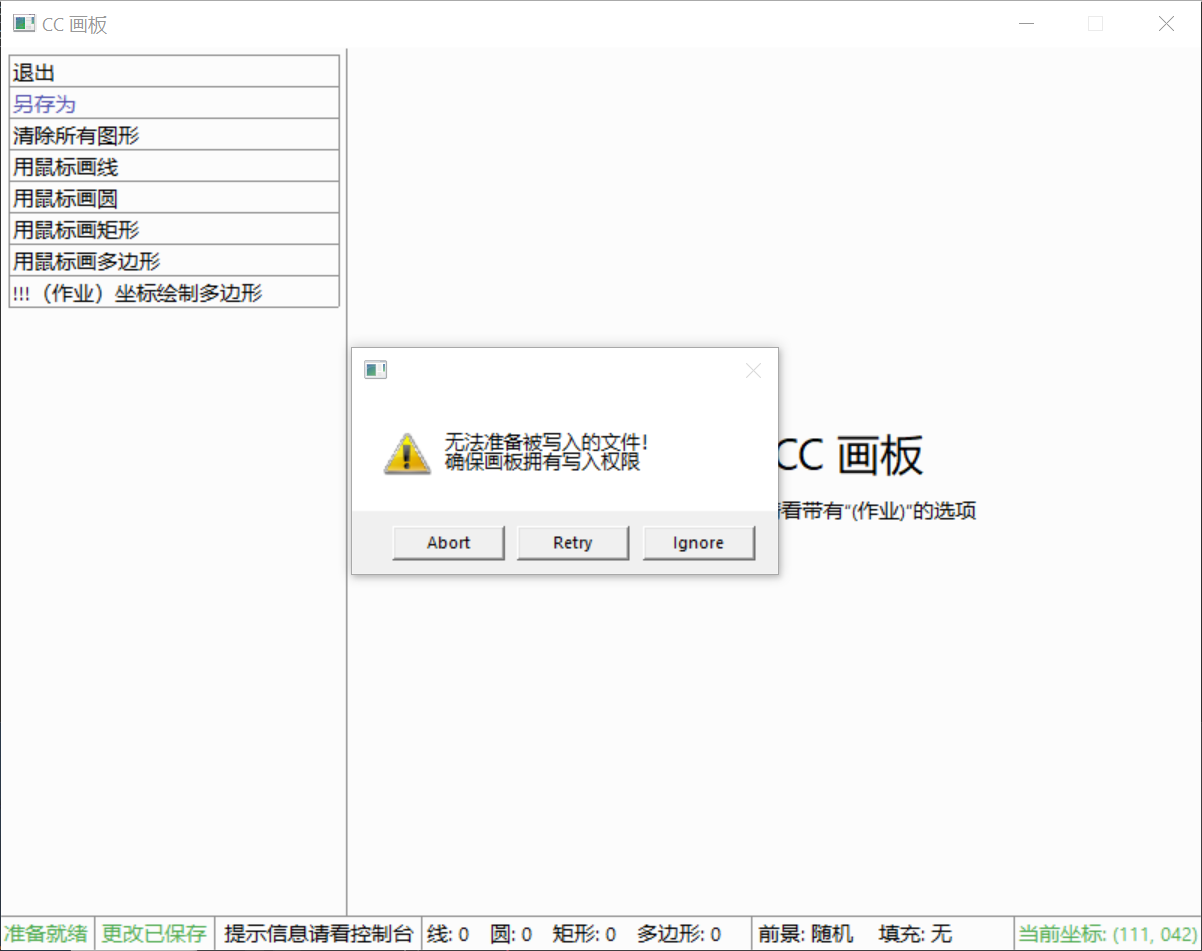


### 异常情况

#### 读取了其他文件或读取存档失败



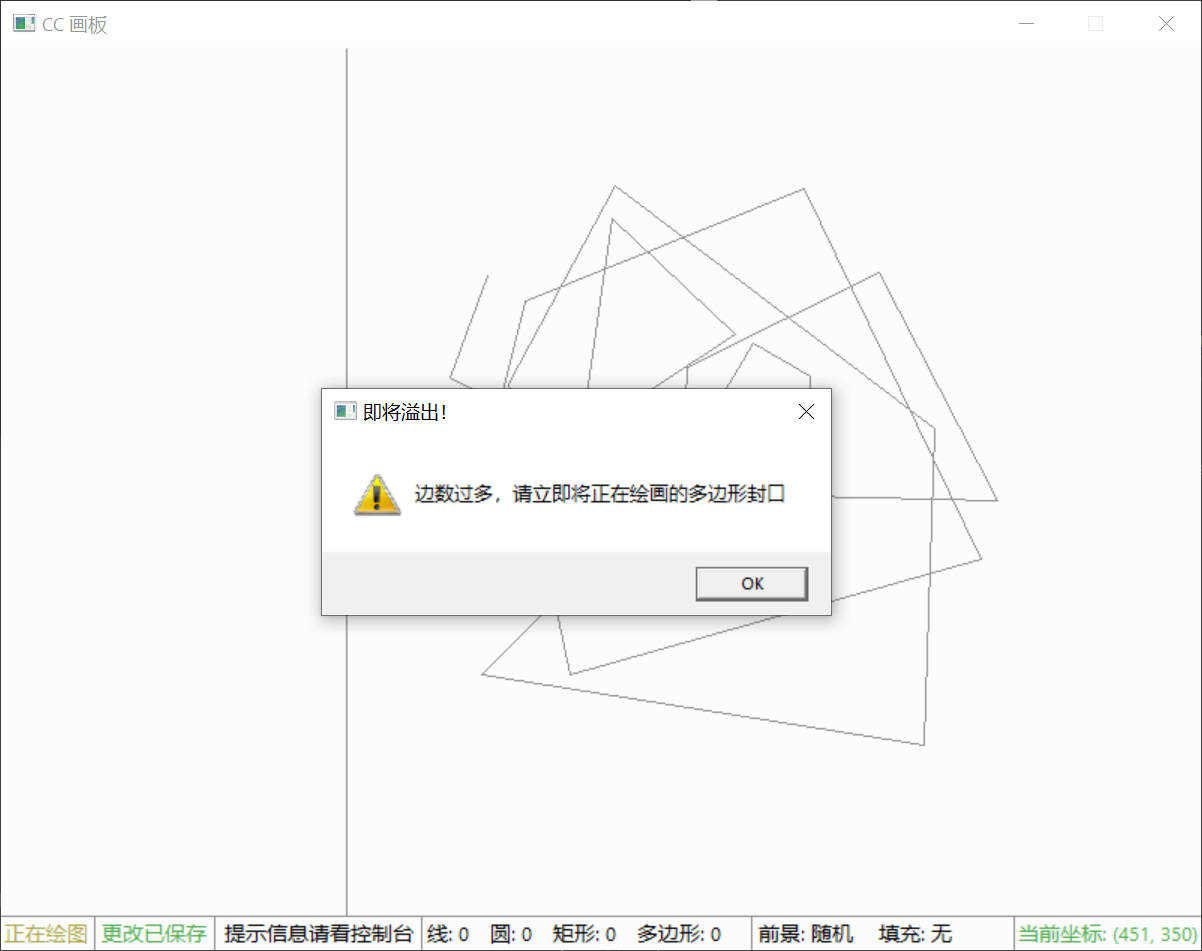
#### 保存存档失败



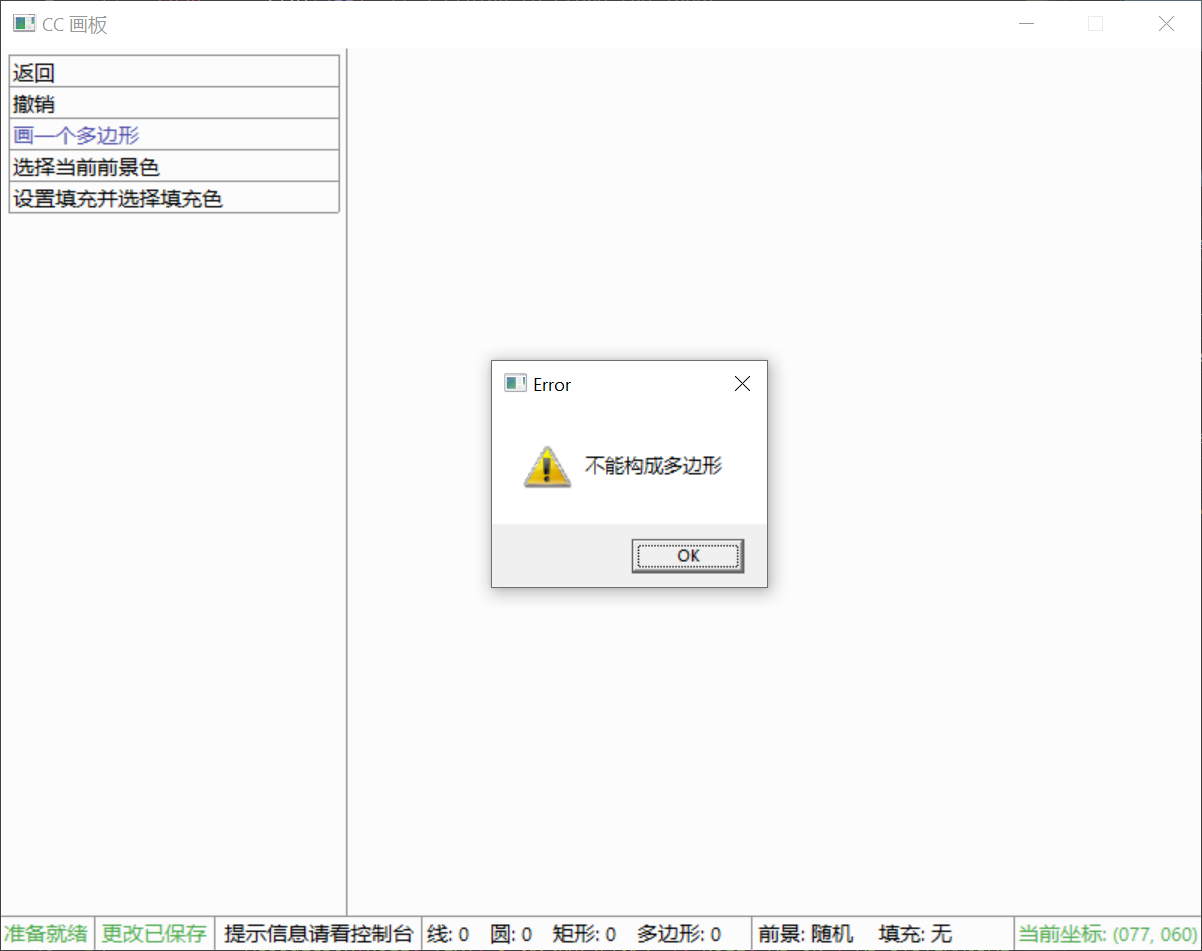
#### 图形数量溢出



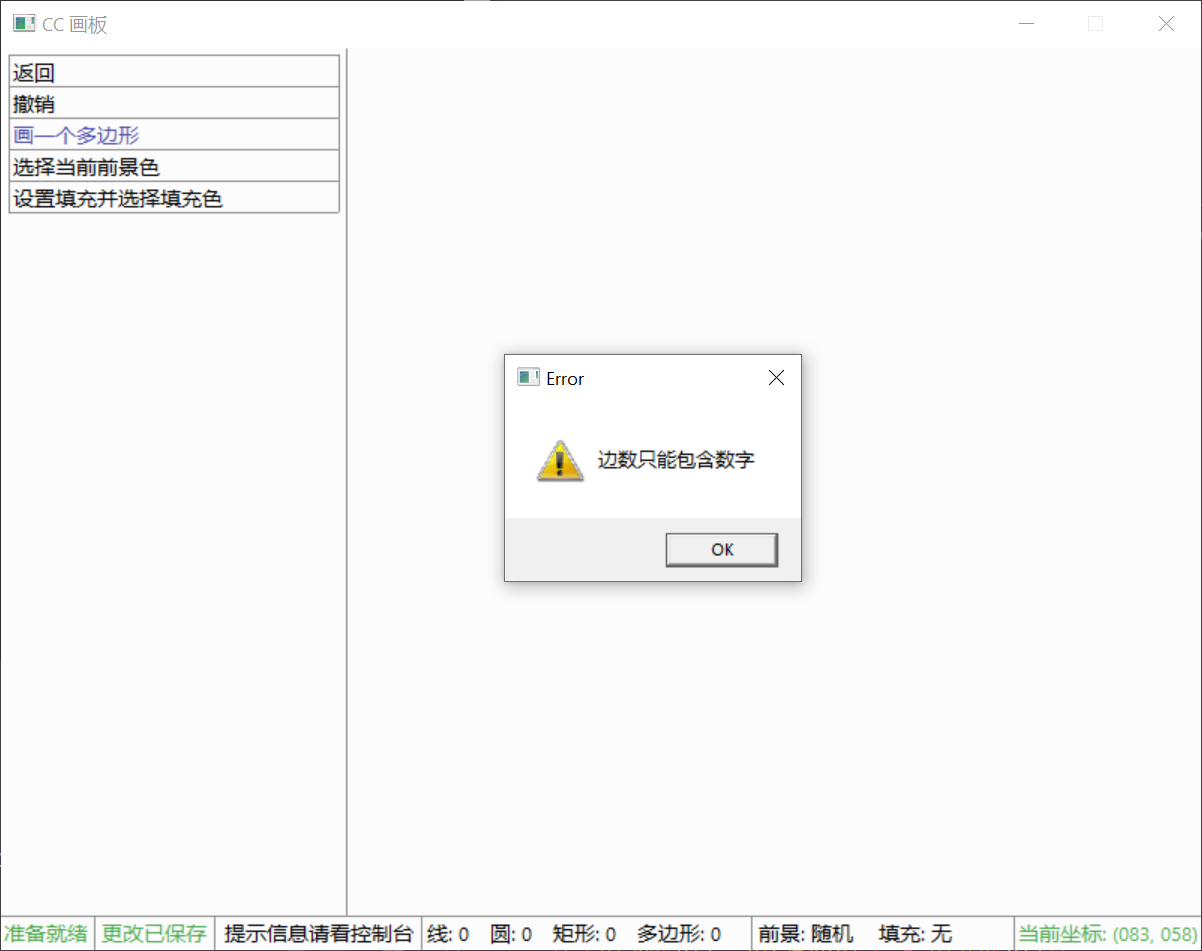
#### 多边形边数溢出



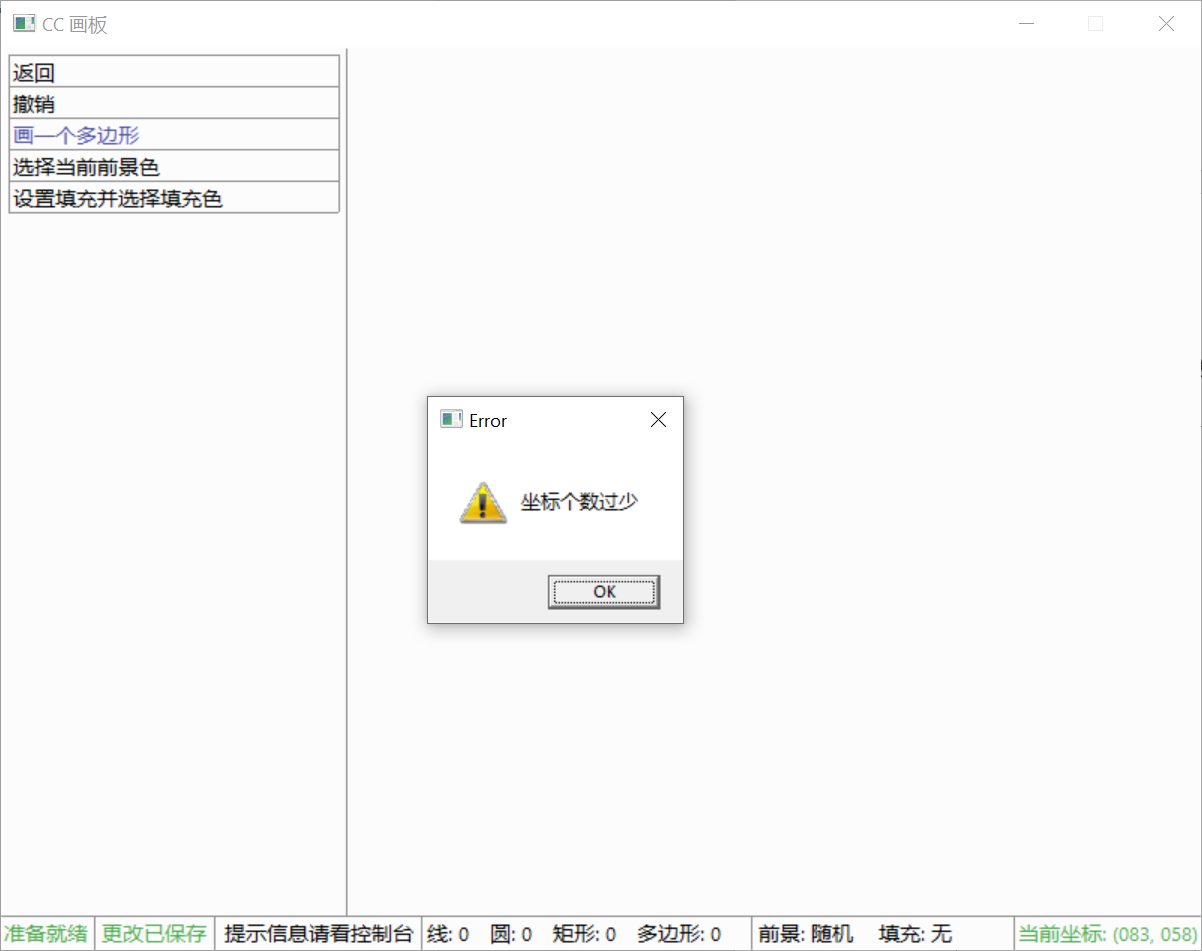
#### 输入的多边形边数过少

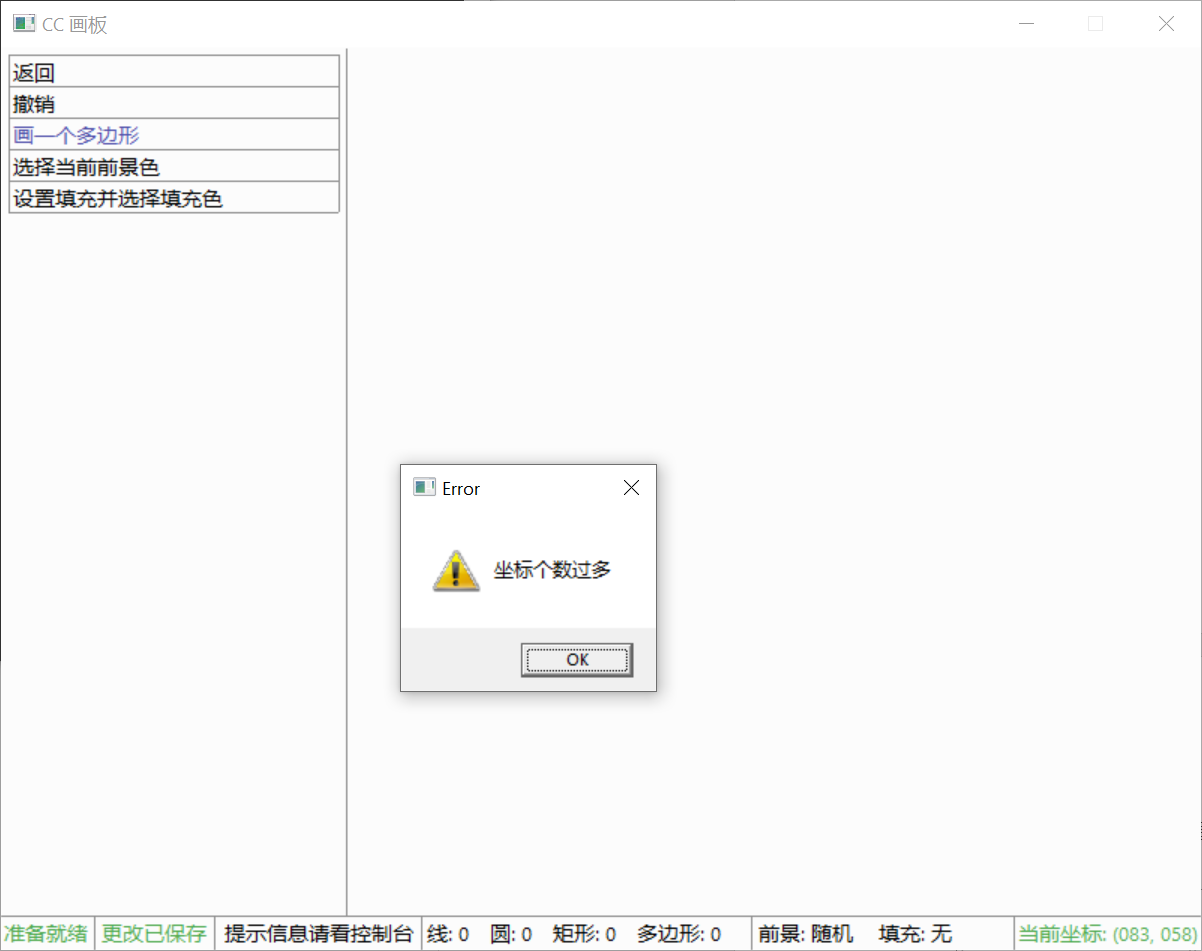


#### 输入的多边形边数或坐标中有非数字

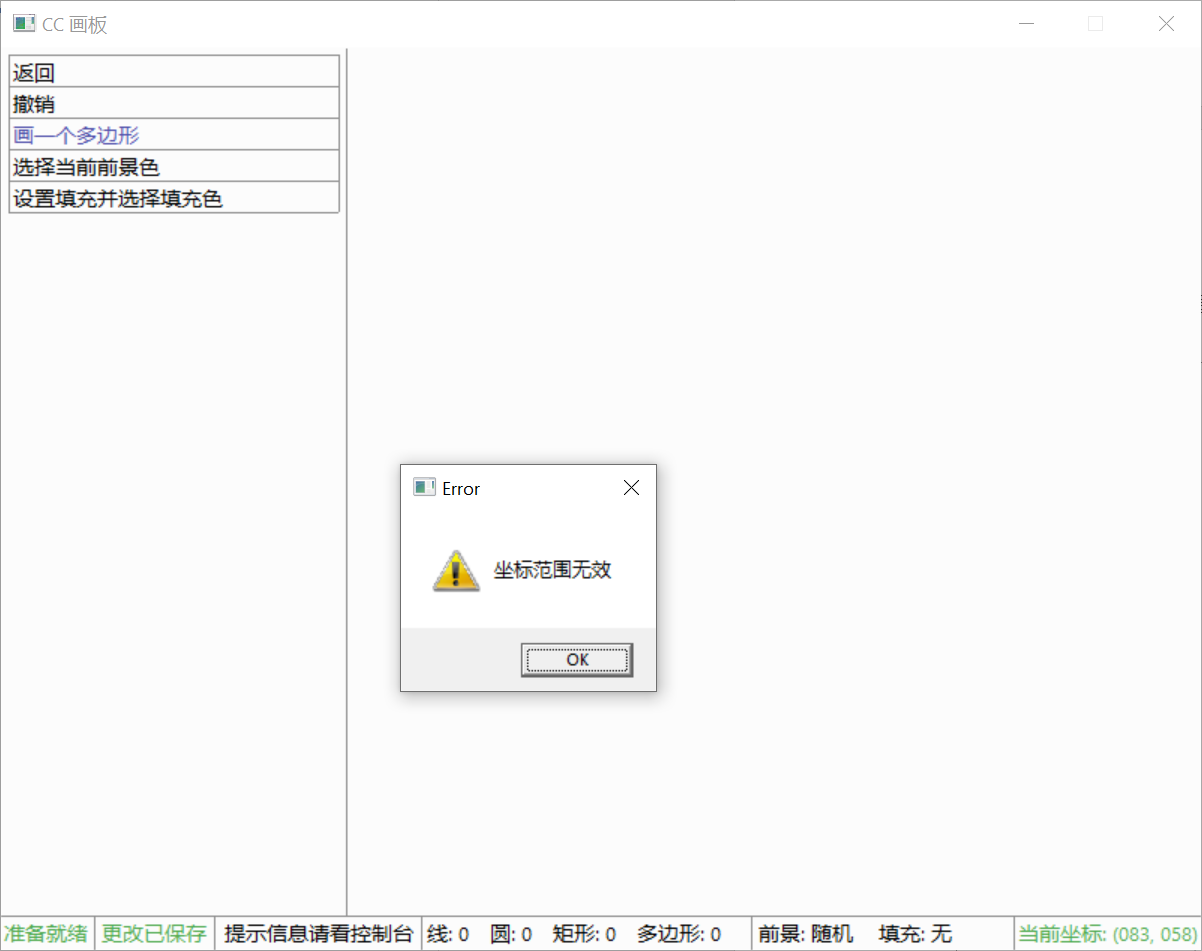


#### 输入的多边形坐标个数过多或过少

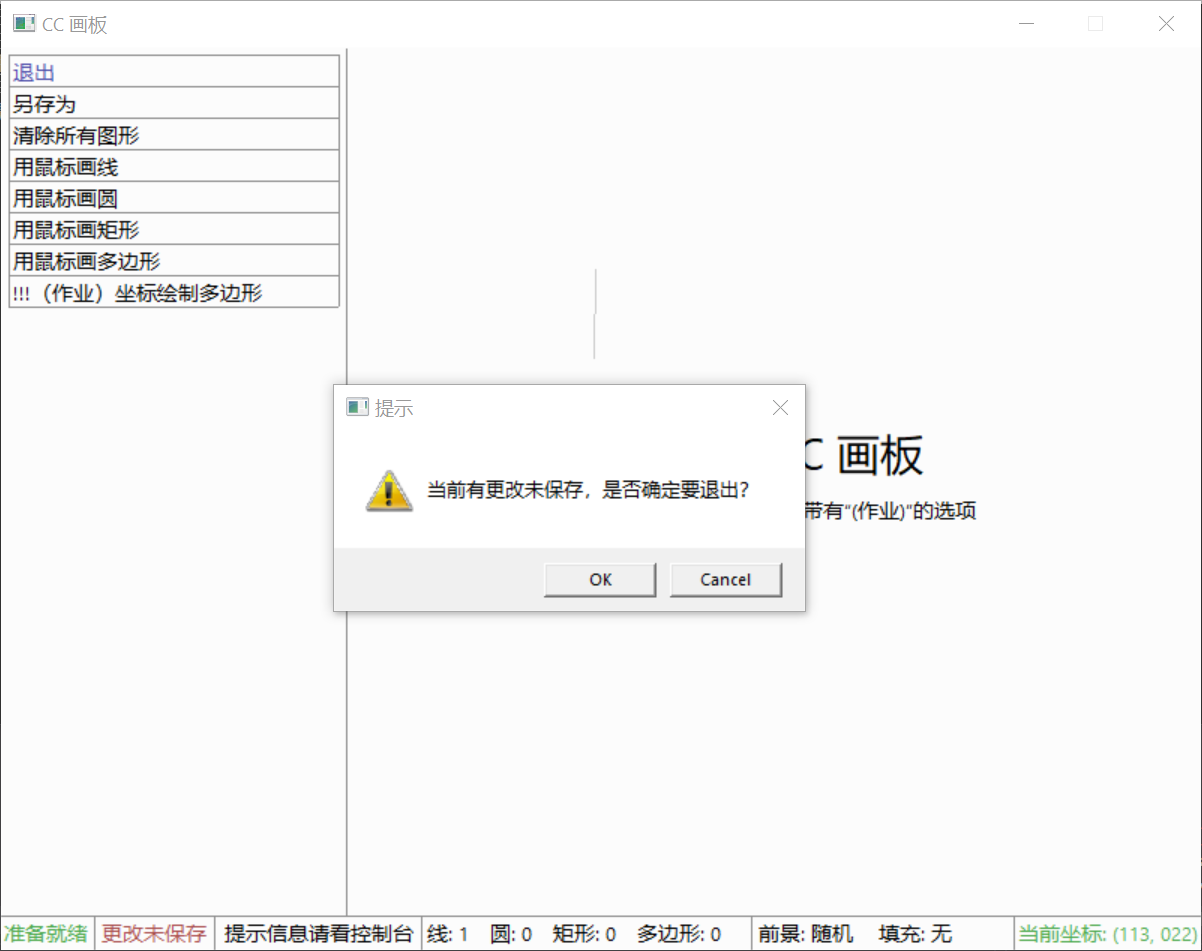




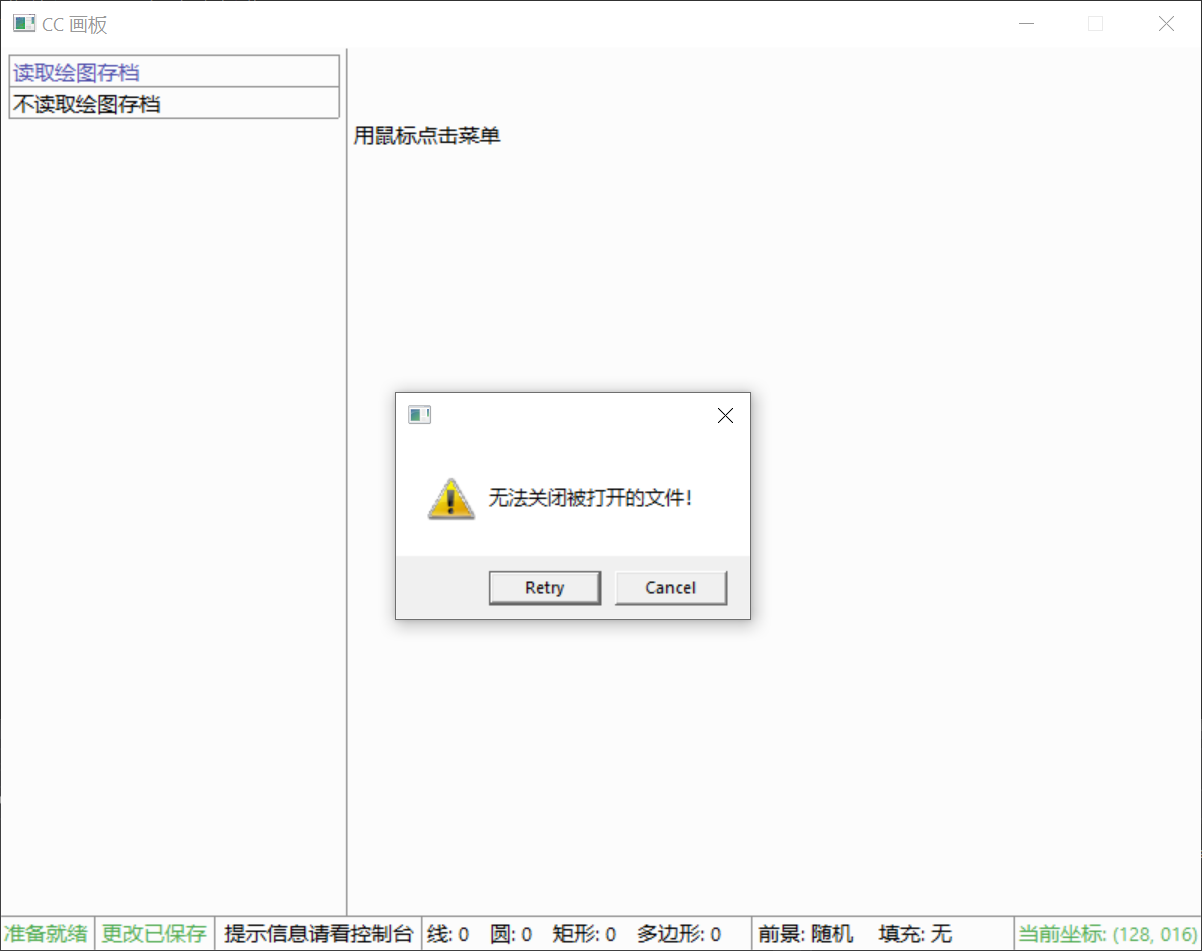
#### 输入的多边形坐标范围溢出



#### 用户在未保存文件的情况下尝试关闭程序



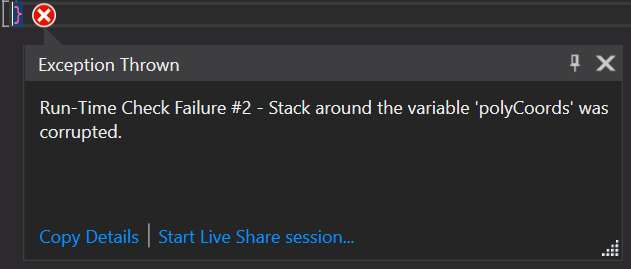
#### 打开文件后关闭异常



## 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）

### Run-Time Check Failure



此问题多为数组越界等其他非法行为导致。

### Initialization skipped by ‘case’label



此问题多为变量的初始化在case内部被跳过导致。

### Uninitialized variable

在使用一个变量之前初始化它。