**《计算机图形学》**

**实验报告4**

**班 级 数172**

**学 号 10172911**

**姓 名 梁天一**

**批阅教师 赵宏庆**

实验4 实现简单的直线作图

# 实验目的及要求

目的：掌握VC++基本的作图方法。

要求：1 实现中点画圆法

2 实现内点表示的四连通区域的递归填充算法---FloodFill4法 P43

3 实现边界表示的四连通区域的递归填充算法---BoundaryFill4法 P43

# 实验步骤

## 中点画圆法

### 代码：

void CMFCApplication5View::CirclePoints(CDC \*pDC, int x, int y, int color)

{

pDC->SetPixel(100 + x, 100 + y, color); pDC->SetPixel(100 + y, 100 + x, color);

pDC->SetPixel(100 - x, 100 + y, color); pDC->SetPixel(100 + y, 100 + -x, color);

pDC->SetPixel(100 + x, 100 - y, color); pDC->SetPixel(100 - y, 100 + x, color);

pDC->SetPixel(100 - x, 100 - y, color); pDC->SetPixel(100 - y, 100 - x, color);

}

void CMFCApplication5View::OnDraw(CDC\* pDC)

{

CMFCApplication5Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

MidPoints(pDC, 50, RGB(255, 0, 0));

//DrawScene();

//DDALine(pDC, 100, 100, 200, 200, RGB(255, 0, 0));

//Bresenham(pDC, 100, 100, 800, 200, RGB(255, 0, 0));

//wglMakeCurrent(pDC->m\_hDC, NULL);

}

void CMFCApplication5View::MidPoints(CDC \*pDC, int r, int color)

{

int x, y;

int e;

x = 0; y = r; e = 1 - r;

CirclePoints(pDC, x, y, color);

while (x <= y)

{

if (e < 0)

e += 2 \* x + 3;

else

{

e += 2 \* (x - y) + 5;

y--;

}

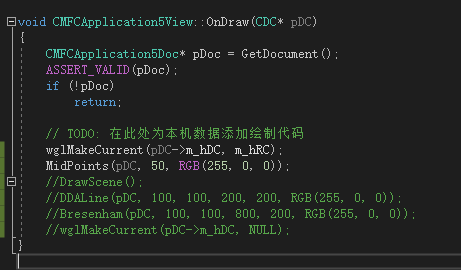
x++;

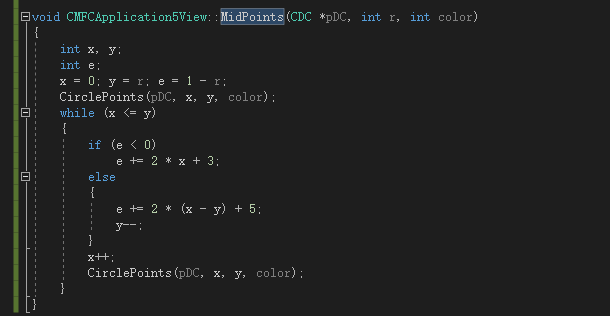
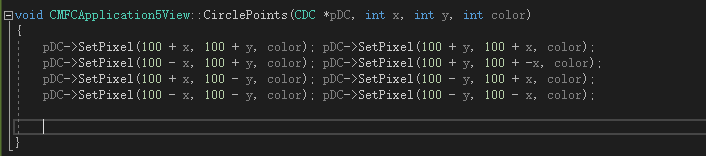
CirclePoints(pDC, x, y, color);

}

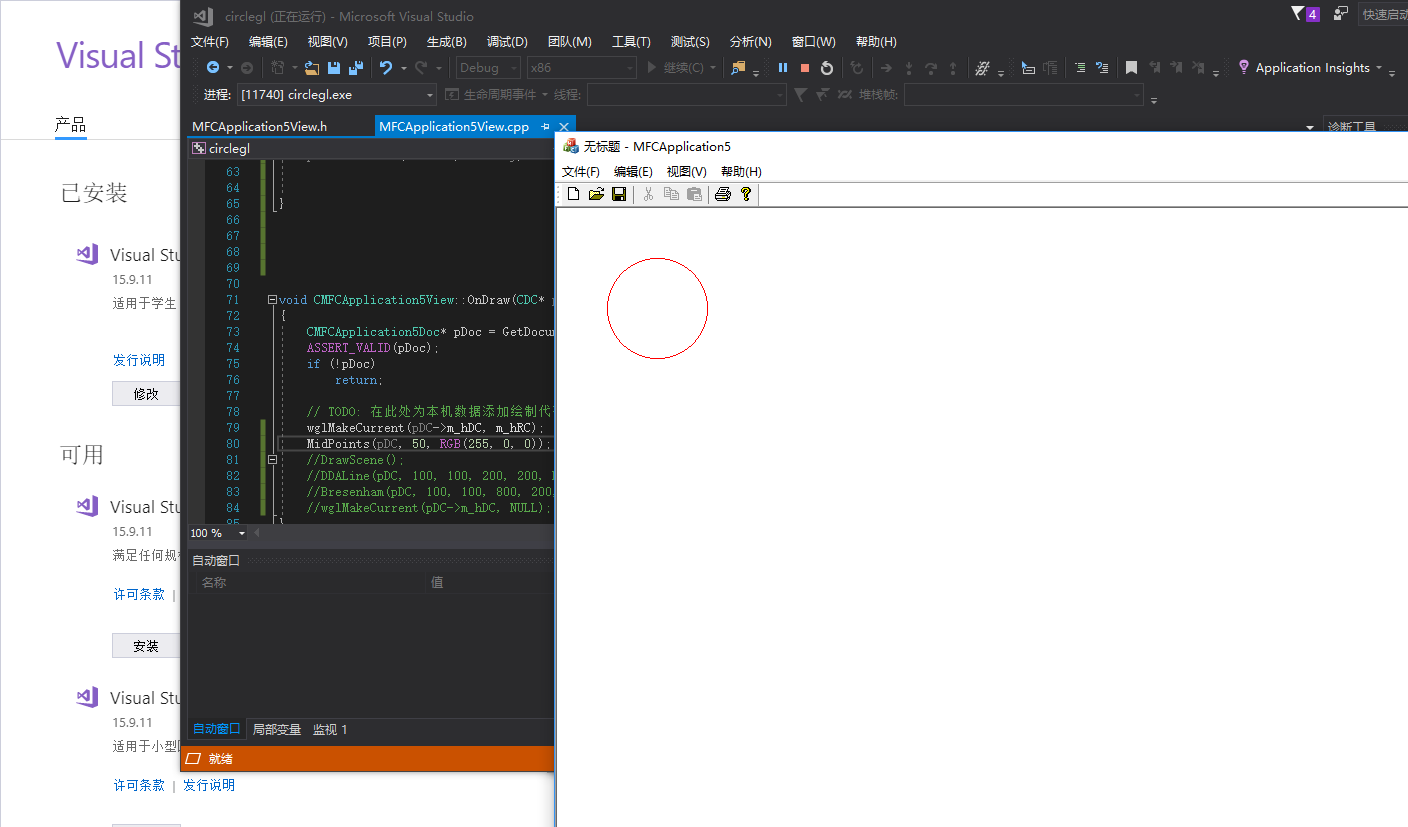
}

### 绘图界面：



### 运行结果：



## FloodFill4法

### 代码：

void CMFCApplication5View::OnDraw(CDC\* pDC)

{

CMFCApplication5Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

//MidPoints(pDC, 50, RGB(255, 0, 0));

pDC->Rectangle(50, 50, 100, 100); //创建一个矩形区域以用来填充

FloodFill4(pDC, 60, 70, RGB(255, 255, 255), RGB(255, 0, 0));//将原来的白色矩形的内部填充为红色

//DrawScene();

//DDALine(pDC, 100, 100, 200, 200, RGB(255, 0, 0));

//Bresenham(pDC, 100, 100, 800, 200, RGB(255, 0, 0));

//wglMakeCurrent(pDC->m\_hDC, NULL);

}

void CMFCApplication5View::FloodFill4(CDC \*pDC, int x, int y, int oldcolor, int newcolor)

{

if (pDC->GetPixel(x, y) == oldcolor)

{

pDC->SetPixel(x, y, newcolor);

FloodFill4(pDC, x, y + 1, oldcolor, newcolor);

FloodFill4(pDC, x, y - 1, oldcolor, newcolor);

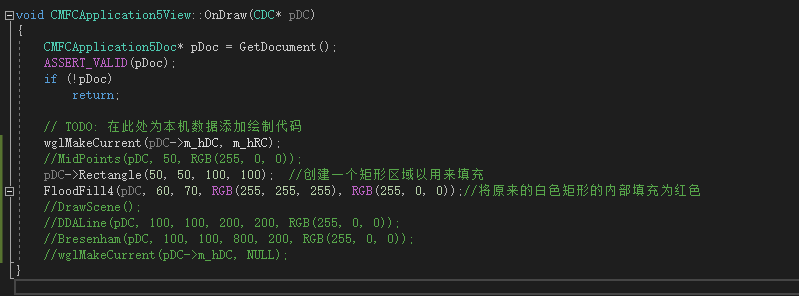
FloodFill4(pDC, x - 1, y, oldcolor, newcolor);

FloodFill4(pDC, x + 1, y, oldcolor, newcolor);

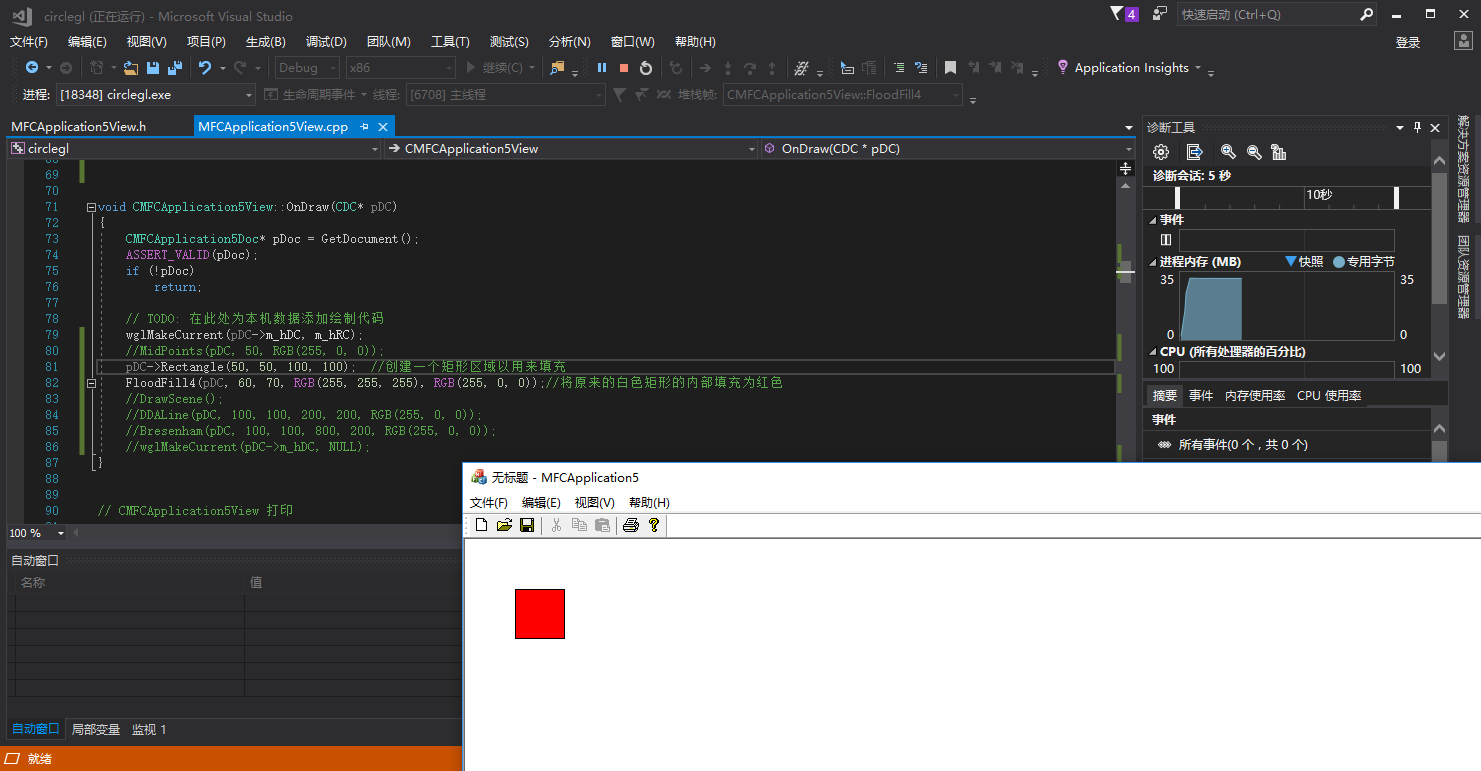
}

}

### 绘图界面：



### 运行结果：



## BoundaryFill4法

### 代码：

void CMFC\_OPEGNGLView::OnDraw(CDC\* pDC)

{

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

pDC->Rectangle(80, 80, 100, 100);

BoundaryFill4(pDC, 89, 90, RGB(0, 0, 0), RGB(255, 0, 0));

CMFC\_OPEGNGLDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//wglMakeCurrent(pDC->m\_hDC, m\_hRC);

//DrawScene();

//wglMakeCurrent(pDC->m\_hDC, NULL);

// TODO: add draw code for native data here

//pDC->MoveTo(100, 100);

//pDC->LineTo(200, 200);

}

void CMFC\_OPEGNGLView::BoundaryFill4(CDC\*pDC, int x, int y, int boundarycolor, int newcolor)

{

int color = pDC->GetPixel(x, y);

if(color != newcolor && color != boundarycolor)

{

pDC->SetPixel(x, y, newcolor);

BoundaryFill4(pDC, x, y + 1, boundarycolor, newcolor);

BoundaryFill4(pDC, x, y - 1, boundarycolor, newcolor);

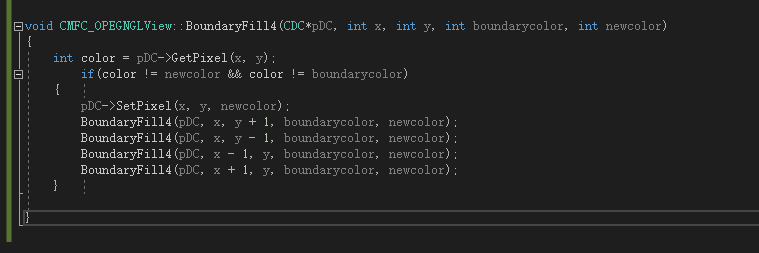
BoundaryFill4(pDC, x - 1, y, boundarycolor, newcolor);

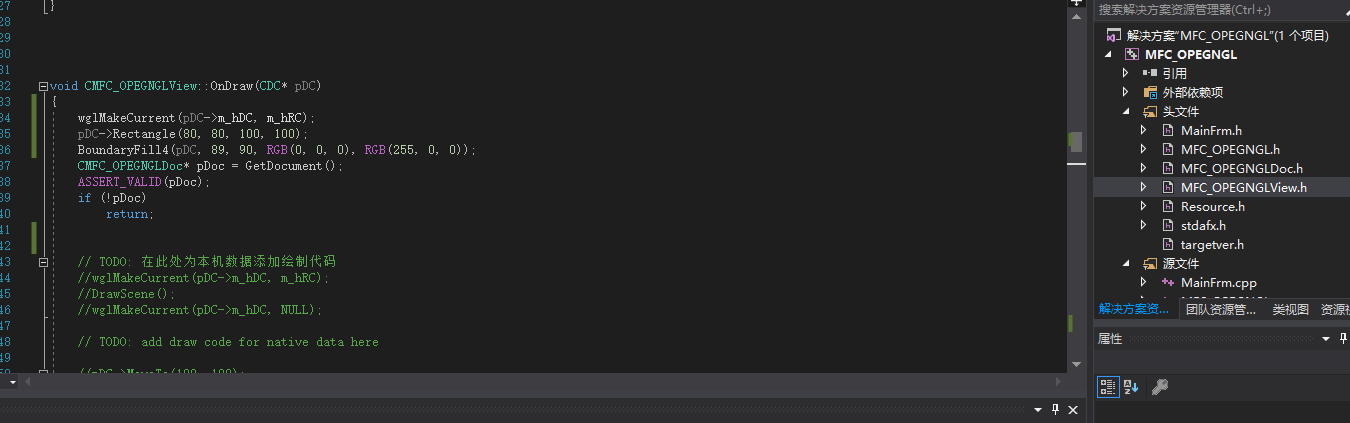
BoundaryFill4(pDC, x + 1, y, boundarycolor, newcolor);

}

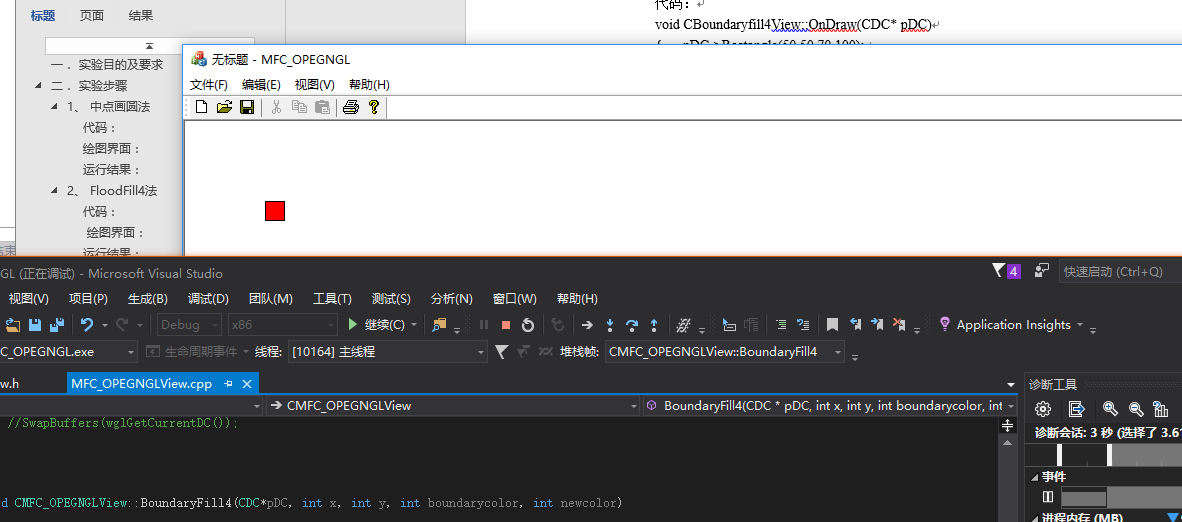
}

### 绘图界面：





### 运行结果：



# 核心算法

## 中点画圆法

void CMFCApplication5View::CirclePoints(CDC \*pDC, int x, int y, int color)

{

pDC->SetPixel(100 + x, 100 + y, color); pDC->SetPixel(100 + y, 100 + x, color);

pDC->SetPixel(100 - x, 100 + y, color); pDC->SetPixel(100 + y, 100 + -x, color);

pDC->SetPixel(100 + x, 100 - y, color); pDC->SetPixel(100 - y, 100 + x, color);

pDC->SetPixel(100 - x, 100 - y, color); pDC->SetPixel(100 - y, 100 - x, color);

}

void CMFCApplication5View::OnDraw(CDC\* pDC)

{

CMFCApplication5Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

MidPoints(pDC, 50, RGB(255, 0, 0));

//DrawScene();

//DDALine(pDC, 100, 100, 200, 200, RGB(255, 0, 0));

//Bresenham(pDC, 100, 100, 800, 200, RGB(255, 0, 0));

//wglMakeCurrent(pDC->m\_hDC, NULL);

}

void CMFCApplication5View::MidPoints(CDC \*pDC, int r, int color)

{

int x, y;

int e;

x = 0; y = r; e = 1 - r;

CirclePoints(pDC, x, y, color);

while (x <= y)

{

if (e < 0)

e += 2 \* x + 3;

else

{

e += 2 \* (x - y) + 5;

y--;

}

x++;

CirclePoints(pDC, x, y, color);

}

}

## FloodFill4法

void CMFCApplication5View::OnDraw(CDC\* pDC)

{

CMFCApplication5Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

//MidPoints(pDC, 50, RGB(255, 0, 0));

pDC->Rectangle(50, 50, 100, 100); //创建一个矩形区域以用来填充

FloodFill4(pDC, 60, 70, RGB(255, 255, 255), RGB(255, 0, 0));//将原来的白色矩形的内部填充为红色

//DrawScene();

//DDALine(pDC, 100, 100, 200, 200, RGB(255, 0, 0));

//Bresenham(pDC, 100, 100, 800, 200, RGB(255, 0, 0));

//wglMakeCurrent(pDC->m\_hDC, NULL);

}

void CMFCApplication5View::FloodFill4(CDC \*pDC, int x, int y, int oldcolor, int newcolor)

{

if (pDC->GetPixel(x, y) == oldcolor)

{

pDC->SetPixel(x, y, newcolor);

FloodFill4(pDC, x, y + 1, oldcolor, newcolor);

FloodFill4(pDC, x, y - 1, oldcolor, newcolor);

FloodFill4(pDC, x - 1, y, oldcolor, newcolor);

FloodFill4(pDC, x + 1, y, oldcolor, newcolor);

}

}

## BoundaryFill4法

void CMFC\_OPEGNGLView::OnDraw(CDC\* pDC)

{

wglMakeCurrent(pDC->m\_hDC, m\_hRC);

pDC->Rectangle(80, 80, 100, 100);

BoundaryFill4(pDC, 89, 90, RGB(0, 0, 0), RGB(255, 0, 0));

CMFC\_OPEGNGLDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//wglMakeCurrent(pDC->m\_hDC, m\_hRC);

//DrawScene();

//wglMakeCurrent(pDC->m\_hDC, NULL);

// TODO: add draw code for native data here

//pDC->MoveTo(100, 100);

//pDC->LineTo(200, 200);

}

void CMFC\_OPEGNGLView::BoundaryFill4(CDC\*pDC, int x, int y, int boundarycolor, int newcolor)

{

int color = pDC->GetPixel(x, y);

if(color != newcolor && color != boundarycolor)

{

pDC->SetPixel(x, y, newcolor);

BoundaryFill4(pDC, x, y + 1, boundarycolor, newcolor);

BoundaryFill4(pDC, x, y - 1, boundarycolor, newcolor);

BoundaryFill4(pDC, x - 1, y, boundarycolor, newcolor);

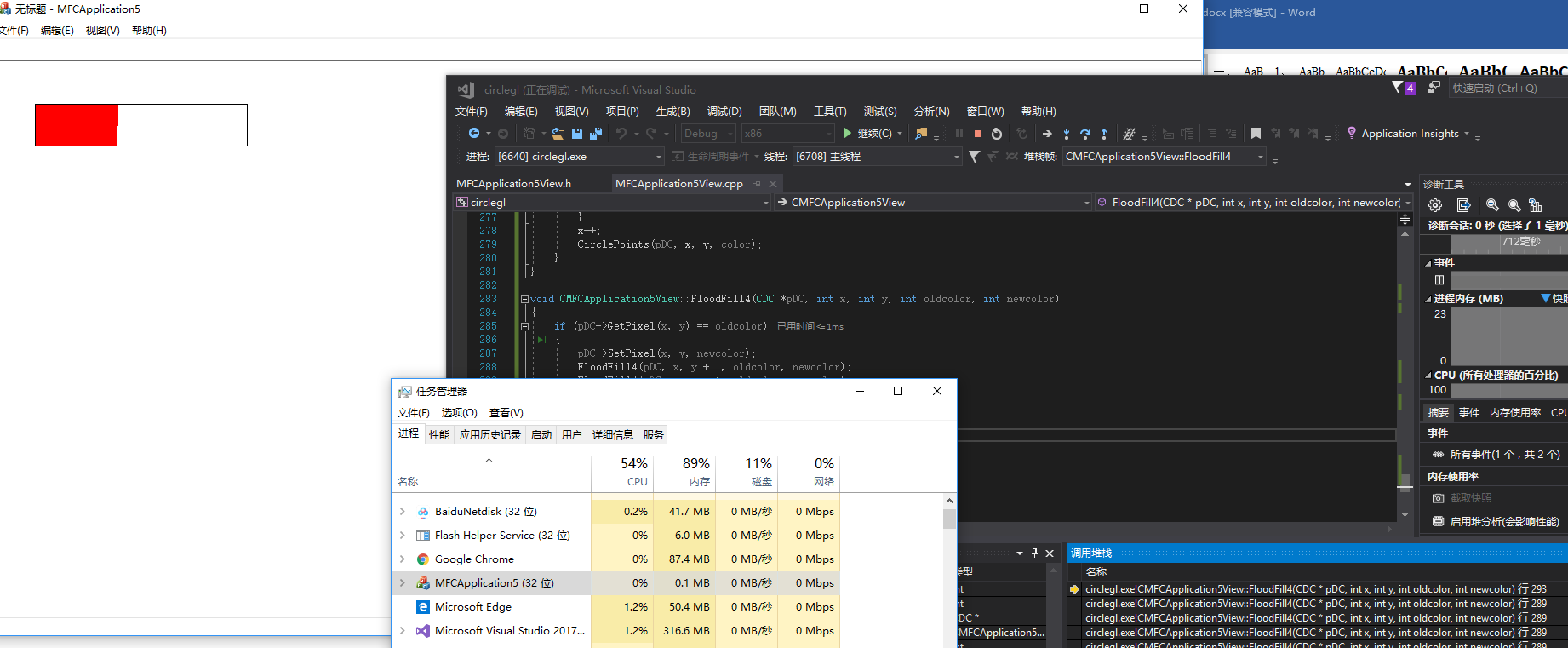
BoundaryFill4(pDC, x + 1, y, boundarycolor, newcolor);

}

}

# 程序调试及运行结果的自我分析与自我评价

不要在填充区域时设置区域过大，否则会内存溢出，比如这样



在进行中点画圆法的时候需要将圆心设置在图中央以显示完整的圆；

在boundaryfill时要把初始点设置在设置区域里面，不然就会根据函数往迭代向着下边一直延伸，内存爆掉