《计算机图形学原理实践》

实验报告

|  |  |  |
| --- | --- | --- |
| 实验名称 | ： | 动态绘制Bezier曲线 |
|  |  |  |
| 姓 名 | ： |  |
| 学 号 | ： |  |
| 专业班级 | ： |  |
| 实验时间 | ： |  |

西南科技大学计算机科学与技术学院

1. **实验目的**

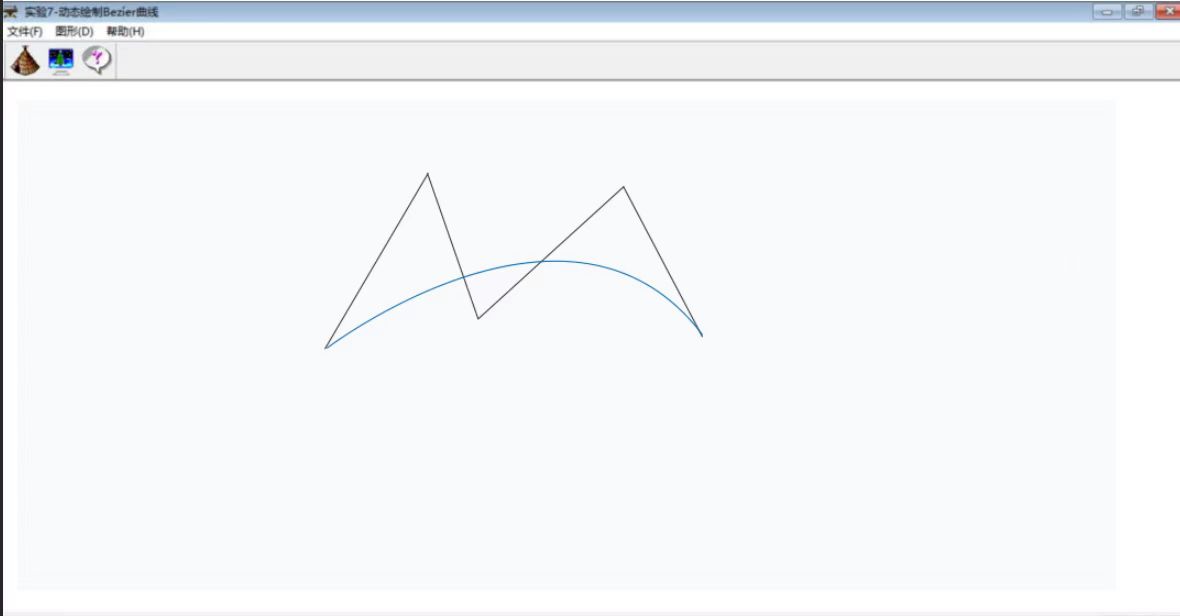
1、掌握直线的参数表示法。

2、掌握德卡斯特利奥算法的几何意义。  
3、掌握绘制二维Bezier。

**二、实验步骤**  
第一步：在屏幕是使用鼠标左键指定控制点，将控制点坐标存储在数组P中

第二步：计算读入的顶点个数CtrPointNum,Bezier曲线的阶次n=CtrPointNum-1。  
第三步：将i从0循环到n，依次使用直线段连接P[i]与P[i+1]为控制多边形，以每个控制点坐标为圆心绘制半径为两个像素的灰色实心圆。  
第四步：右鼠标调用DrawBezier()函数绘制Bezier()曲线。  
第五步：在DrawBezier()函数中，创建二维动态数组PP,并且使用CDC类的成员函数Moveto()将直线段的当前位置置为P[0]。  
第六步：对参数t从0按到1/100的步长循环到1。  
第七步：调用 de casteliau 函数计算每个t对应的pp。  
第八步：当t<=1.0时，使用CDC类成员函数Line To()连接曲线上的各点。否则，程序结束。

**三、实验结果**



1. **实验总结**

本次实验根据德卡斯特利奥算法绘制动态Bezier曲线，n+1个顶点构成的控制多边形产生n次Bezier曲线，曲线的起点和终点与多边形的起点与终点重合，且控制多边形的第一条边和最后一条边表示曲线在起点和终点的切矢量方向。曲线的形状趋于控制多边形的形状。总的来说，这次实验让我收获良多。

**附录：源代码（关键代码）**

// TestView.cpp : CTestView 类的实现

//

#include "stdafx.h"

// SHARED\_HANDLERS 可以在实现预览、缩略图和搜索筛选器句柄的

// ATL 项目中进行定义，并允许与该项目共享文档代码。

#ifndef SHARED\_HANDLERS

#include "Test.h"

#endif

#include "TestDoc.h"

#include "TestView.h"

#include "math.h"//包含数学头文件

#define PI 3.1415926//PI的宏定义

#define Round(d) int(floor(d+0.5))//四舍五入宏定义

#define N\_MAX\_POINT 21//控制多边形的最大顶点数

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CTestView

IMPLEMENT\_DYNCREATE(CTestView, CView)

BEGIN\_MESSAGE\_MAP(CTestView, CView)

// 标准打印命令

ON\_COMMAND(ID\_FILE\_PRINT, &CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_DIRECT, &CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_PREVIEW, &CView::OnFilePrintPreview)

ON\_COMMAND(IDM\_DRAWPIC, &CTestView::OnDrawpic)

ON\_WM\_LBUTTONDOWN()

// ON\_WM\_LBUTTONUP()

ON\_WM\_RBUTTONDOWN()

END\_MESSAGE\_MAP()

// CTestView 构造/析构

CTestView::CTestView()

{

// TODO: 在此处添加构造代码

P=new CPoint[N\_MAX\_POINT];

bFlag=FALSE;

CtrlPointNum=0;

}

CTestView::~CTestView()

{

if(P!=NULL)

{

delete []P;

P=NULL;

}

}

BOOL CTestView::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: 在此处通过修改

// CREATESTRUCT cs 来修改窗口类或样式

return CView::PreCreateWindow(cs);

}

// CTestView 绘制

void CTestView::OnDraw(CDC\* /\*pDC\*/)

{

CTestDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

}

// CTestView 打印

BOOL CTestView::OnPreparePrinting(CPrintInfo\* pInfo)

{

// 默认准备

return DoPreparePrinting(pInfo);

}

void CTestView::OnBeginPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: 添加额外的打印前进行的初始化过程

}

void CTestView::OnEndPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: 添加打印后进行的清理过程

}

// CTestView 诊断

#ifdef \_DEBUG

void CTestView::AssertValid() const

{

CView::AssertValid();

}

void CTestView::Dump(CDumpContext& dc) const

{

CView::Dump(dc);

}

CTestDoc\* CTestView::GetDocument() const // 非调试版本是内联的

{

ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CTestDoc)));

return (CTestDoc\*)m\_pDocument;

}

#endif //\_DEBUG

// CTestView 消息处理程序

void CTestView::OnDrawpic()

{

// TODO: Add your command handler code here

MessageBox(CString("左键绘制控制多边形，右键绘制曲线"),CString("提示"),MB\_OK);

RedrawWindow();

bFlag=TRUE;

CtrlPointNum=0;

}

long CTestView::DeCasteliau(double t, long\* p)

{

double P[N\_MAX\_POINT][N\_MAX\_POINT];

int n = CtrlPNum - 1;

for (int k = 0;k <= n;k++)

{

P[0][k] = p[k];

}

for (int r = 1;r <= n;r++)

{

for (int i = 0;i <= n - r;i++)

{

P[r][i] = (1 - t) \* P[r - 1][i] + t \* P[r - 1][i + 1];

}

}

return (long(P[n][0]));

}

void CTestView::DrawBezier()//绘制Bezier曲线

{

CDC\* pDC = GetDC();

double delt = 1.0 / 50000;

CPoint p;

long px[N\_MAX\_POINT], py[N\_MAX\_POINT];

for (int k = 0;k <= n;k++)

{

px[k] = pt[k].x;

py[k] = pt[k].y;

}

for (double t = 0;t <= 1;t += delt)

{

p.x = DeCasteliau(t, px);

p.y = DeCasteliau(t, py);

pDC->SetPixel(p.x, p.y, RGB(0, 0, 255));

}

ReleaseDC(pDC);

}

void CTestView::deCasteljau(double t, CP2\* p)//de Casteljau函数

{

}

void CTestView::DrawCtrlPolygon()//绘制控制多边形

{

CDC \*pDC=GetDC();

CBrush NewBrush,\*pOldBrush;

pOldBrush=(CBrush\*)pDC->SelectStockObject(GRAY\_BRUSH);//选择灰色库画刷

for(int i=0;i<=n;i++)

{

if(0==i)

{

pDC->MoveTo(P[i]);

pDC->Ellipse(P[i].x-2,P[i].y-2,P[i].x+2,P[i].y+2);

}

else

{

pDC->LineTo(P[i]);

pDC->Ellipse(P[i].x-2,P[i].y-2,P[i].x+2,P[i].y+2);

}

}

pDC->SelectObject(pOldBrush);

ReleaseDC(pDC);

}

void CTestView::OnLButtonDown(UINT nFlags, CPoint point)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

if(bFlag)

{

P[CtrlPointNum].x=point.x;

P[CtrlPointNum].y=point.y;

if(CtrlPointNum<N\_MAX\_POINT-1)

CtrlPointNum++;

else

bFlag=FALSE;

n=CtrlPointNum-1;

DrawCtrlPolygon();

}

CView::OnLButtonDown(nFlags, point);

}

void CTestView::OnRButtonDown(UINT nFlags, CPoint point)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

bFlag=FALSE;

if(0!=CtrlPointNum)

DrawBezier();

CView::OnRButtonDown(nFlags, point);

}