第一章：P56

1. 列出在你过去学习工作中用过与计算机图形学有关的程序

c语言：

#include <graphics.h>

main()

{

int graphdriver = VGA, graphmode=VGAHI;

initgraph(&graphdriver,&graphmode,””);

setbkcolor(BLUE);

setcolor(WHITE);

setfillstyle(1,LIGHTRED);

bar3d(100,200,400,350,100,1);

floodfill(450,300,WHITE);

floodfill(250,450,WHITE);

setcolor(LIGHTGREEN);

rectangle(450,400,500,450);

floodfill(470,420,LIGHTGREEN);

getch();

closegraph();

}

JAVA语言：

例1、画点

Import java.io.\*;

Class point

{

int ax;

int ay;

int bx;

int by;

public point(int ax, int ay, int bx, int by)

{

float k ; //计算斜率

float b;

k=(by-ay)/(bx-ax);

b=ay-ax\*k;

system.out.println(“直线的方程为：y=”+k+”x”+”+”+b);

}

}

例2、画矩形

class DrawPanel extends Jpanel

{

public void paint(Graphics g)

{

super.paint(g);

Graphics2D g2= (Graphics 2D);

Double leftx=200;

Double topy=200;

Double width=300;

Double height=250;

Rectangle2D rect= new Rectangle2D.double(leftx,topy,width,height);

G2.draw(rect);

}

}

1. 列出你所用过的窗口系统中与观感有关的元素的功能，如图标、滚动棒、菜单等

## 使用滚动条

当文档、网页或图片超出窗口大小时，会出现滚动条，可用于查看当前处于视图之外的信息。下面的图片显示滚动条的组成部分。

## 使用菜单

大多数程序包含几十个甚至几百个使程序运行的*命令*（操作）。很多这些命令是组织在*菜单*下面。就像饭馆的菜单一样，程序菜单显示选择列表。为了使屏幕整齐，会隐藏这些菜单，只有在标题栏下的*菜单栏*中单击菜单标题之后才会显示菜单。例如，单击“画图”菜单栏中的“图像”可显示“图像”菜单：

1. 列出你所用过的图形输入、显示及输出设备的名称、型号、生产厂商、出厂时间及其主要优缺点。

略

1. 比较个人计算机与工作站的图形功能

个人计算机仅限于符合二维，又是单任务操作方式

工作站可处理二、三维，多任务操作方式

|  |  |  |
| --- | --- | --- |
| 区别 | 个人计算机 | 工作站 |
| 显示分别率 | 640 X 480 | 1024 X 900以上  具有8个以上位面 |
| 显示器尺寸 | 12 ~ 14英寸 | 16、19、27英寸 |
| 图形处理能力 | 符号或二维 | 具有反走样、线和面消隐、光照模型等处理硬件；同时还具有丰富的图形生成和处理软件  主要处理二维或三维图形、图象 |
| 计算机性能 | 主要取决于微处理器的性能 | 具有更强的处理功能，在操作系统、页面虚拟存储器和主要用途都有所不同 |
| 操作方式 | 单任务 | 多任务、多进程 |

* 个人计算机的主要功能是字符处理，而工作站不仅有字符处理功能，还有较强的图形处理功能
* 个人计算机的显示分辨率较小，一般在640X480的图形处理符号或二维图形；而工作站为了满足强大的图形处理，显示分辨率为一般为1024x1024的二维或三维图形
* 个人计算机的显示器相对工作站较小
* 个人计算机的图形系统是由个人计算机加上图形输入输出设备和有关的图形支撑软件集成起来的系统，其性能取决于个人计算机所采用的微处理器芯片，个人计算机的图形功能由于受到软件和硬件的限制，只适合处理比较简单的事情；个人计算机的显示分辨率一般在640X480的图形处理符号或二维图形
* 而工作站在处理器、总线设计、存储器、操作系统等多个方面都有比个人计算机优越的地方，它配有专业的图形输入输出设备，并配有图形处理器以处理大量的复杂的图形运算；它可以处理多任务进程、处理显示分辨率为1024x1024的二维或三维图形、具有高速的科学计算能力、丰富的图形处理、灵活的窗口及网络管理功能的交互式计算机系统，它的图形功能比个人计算机图形系统强德多，但在造价上远高于个人计算机。

6、具有相同分辨率的彩色光栅显示器与黑白光栅在结构上有何区别？

彩色：

对于红、绿、蓝的三个原色有三个位面的帧缓存和三个电子枪，每个位面的帧缓冲对应一个电子枪即对应一种颜色；对每个颜色的电子枪可以通过增加帧缓存位面来提高颜色种类和灰度级，通过三种原色的组合可以产生不同种类的颜色。

彩色光栅显示器主要是有红、绿、蓝的三个原色所组成，每种原色电子枪有8个位面的帧缓存和8位的数模转换器，每种原色有256种亮度，三种原色组合可为16771216种颜色，也可以通过颜色查找表查找，故帧缓存位数至少24位。

黑白：

黑白光栅显示器的帧缓存是一块连续的计算机存储器，每个像素需1位存储器，每个存储器只有0或1两个状态。因此一个位面的帧缓存只能产生黑白图形。可以增加象素点的位面数，通过多个位面显示出多种灰度级。

7、在光栅显示器上显示斜线的45º角时常会发生锯齿状，请考虑减少锯齿状效果的各种方法并说明采用这些方法的代价。

在光栅图形显示器上显示斜线时常会发生锯齿，这是由于直线或多边形边界在光栅图形显示器的对应图形都是由一系列相同亮度的离散象素构成的。这种用离散量表示连续亮引起的失真称为走样，而用于减少或消除这种效果的技术，称为反走样。

一般而言，减少锯齿有三种方法，下面以直线扫描转换为例，分别介绍三种方法：

1. 提高分辨率

假设把显示器的分辨率提高一倍，虽然直线经过2倍的象素，锯齿也会增加一倍。但由于每个锯齿在X方向和Y方向都只有低分辨率的一半，所以效果看起来会好一些，这种改进方法是以4倍的存储器代价和4倍的扫描转换时间获得的。因此增加分辨率是不经济的方法，它只能减轻，不能消除锯齿。

1. 简单的区域取样

在直线扫描算法假定象素是数学上的一个点，象素的颜色是由对应于象素中心的图形中一点的颜色决定的。但是，实际上象素不是一个点，而是一个有限区域。屏幕上所画的直线段不是数学意义上的无宽度的理想线段，而是一个宽度至少为一个象素单位的线条。因此，把屏幕上的直线看成是长方条形更为合理。在绘制直线条时，所有与该长方条相交的象素都采用适当的宽度给予显示。这要求显示器各象素可以用多灰度显示。例，设象素中心是在网格点上的不相交的正方形，象素的灰度与它落在直线条内的面积成正比。在多灰度黑白显示器上，若一个象素整个落在线条上，则将它置成前景色。若一个象素与线条部分相交，根据相交部分的大小来选择不同的灰度，相交部分大的象素前景色成分更多一些，相交部分小的象素前景色成分更少一些。这种方法将产生模糊的边界，以此来减轻锯齿效应。在实际应用中，常采用盒式滤波器

1. 加权区域取样

加权区域取样方法采用更为优化的圆锥形滤波器。圆锥的底圆中心在当前象素中心，底圆半径为一个单位，锥高为1。当直线条经过该象素时，该象素的灰度值是在二者相交区域上对滤波器进行积分的积分值。用这种圆锥形滤波器有如下特点：一是接近理想直线的象素将被分配更多的灰度值。二是相邻两个象素的滤波器相交，所以直线条经过该相交区域时，将对这两个象素分配给适当的灰度值，这有利于缩小直线条上相邻象素的灰度差。

第二章：P128

**1、为什么要制定和采用计算机图形标准？已经ISO批准的计算机图形标准软件有哪些？**

为了提高计算机图形软件、计算机图形的应用软件以及相关软件的编程人员在不同的计算机和图形设备之间的可移植性。

已获ISO批准的计算机图形标准软件有：

* 计算机图形核心系统（GKS）及其语言联编
* 程序员层次交互式图形系统（PHIGS）及其语言联编
* 三维图形核心系统（GKS-3D）及其语言联编
* 计算机图形元文件（CGM）
* 计算机图形接口（CGI）
* 基本图形转换规范（IGES）
* 产品数据转换规范（STEP）等

**2、CGI标准的主要功能是什么？试用CGI中的图形输出功能绘制一副机械零件图。**

CGI的目的是提供控制图形硬件的一种与设备无关的方法，它可以看成是图形设备驱动程序的一种标准。CGI在用户程序和虚拟设备之间，以一种独立于设备的方式提供图形信息的描述和通信，使有经验的用户最大限度地、灵活地直接控制图形设备。它所提供的功能集包括：

* 控制功能集
* 独立于设备的图形对象输出功能集
* 图段功能集
* 输入和应答功能集
* 产生修改、检索和显示以像素数据形式存储的光栅功能集

**3、CGM对文件管理的存储结构是采用何种形式？你认为应用这种结构有什么优缺点？**

采用生成多个与设备无关的图形定义，提供随机存取、传送、简洁定义图象的图形生成元文件的存储结构，它不是应用程序员的标准，而是为系统和系统开发而设计的，与CGI配套供用户使用。

优点是：它具有通用性，即CGM应能广泛适应各种设备、应用系统。例如同一个文件即可在低分辩率的单色图形终端上输出，也可在高分辨率的多笔绘图仪上输出，或在高性能的光栅图形显示器上输出。

这种结构的缺点是：它只是一个静态的图形生成元文件，即它不能产生和定义图形的动态效果，例如不能实现动态的几何变换。

**4、GKS、PHIGS、GI在应用程序中起的作用？试比较它们在输入输出功能上的相同和不同之处？**

GKS在应用程序和图形输入输出设备之间提供了功能接口，包括：控制功能、输出功能、输出属性、变换功能、图段功能、输入功能、询问功能、实用程序、元文件处理和出错处理。

PHIGS向应用程序提供控制图形设备的图形系统接口，能够在系统中高效率地描述应用模型，迅速地修改图形模型的数据；并能够绘制显示修改后的图形模型。

GL是工作站或UNIX上广泛应用的一个工业标准图形程序库，和PHIGS同样是提供用户与程序图形系统接口。包括基本图素、坐标变换、设置属性和显示方式、输入/输出处理、真实图形显示。

相同点：

三个都是提供用户与输入输出设备之间的图形系统接口的标准图形程序库。

不同点：

不同的数据结构

可修改性

属性的存储

输出流水线等

具体而言：

* GKS有6种输入功能和6种输出图素，在输入功能上可对各种设备初始化，设定设备工作方式、确定请求采样和事件输入；在输出功能上，可确定输出图形的类型
* PHIGS的输出流水线有5个坐标系；具有高度的动态性、输出交互性的三维图形，可以在系统中高效率地描述应用模型，迅速修改图形模型的数据，并能绘制显示修改后的图形模型
* GL的输入/输出处理用于启动输入输出设备，并对相应的事件队列进行处理，提供了更丰富的图元，如各种曲面。

6、GKS-3D与PHIGS的主要区别是什么？用GKS-3D输出图形的过程是什么？

主要区别：

1. 数据结构

GKS-3D：提供了单层、平面的图形数据结构

其图段用来表示的是图象信息而不是图形的构造信息，

其图段数据经过坐标规格化变换后，不再是定义该图段的坐标空间的数据

PHIGS：

其结构始终是在造型空间中定义的数据

1. 可修改性

GKS-3D：产生的图段，其内容不能修改，但影响图段整体特征的某些属性，如可见性、可检测性、图段的几何变换等是可以修改

PHIGS：其任何结构，结构中的任何一部分元素则可以在任何时候进行修改

1. 属性的存储

GKS-3D：把图素属性和图素一起存入图形数据结构中，为了修改某图段中某个图素的属性，必须去除该图素的旧属性，重新生成一个新属性

PHIGS：只要当遍历一个结构并要显示该结构时，其中的图素才能变成输出图素，此时，那些属性结构元素是灵活的，图形数据的修改也是容易的。

1. 输出流水线

GKS-3D：采用三种坐标系，用户坐标系、设备坐标系和规格化设备坐标系

PHIGS：采用五种坐标系，造型坐标系、用户坐标系、观察坐标系、规格化的投影（空间）坐标系、设备坐标系

过程：

图素→规格化变换→图段变换→规格化裁减→视图变换→裁减操作和视图映象→工作站裁剪和变换→显示输出

**10、IGES和STEP有什么共同点和不同点？**

共同点：

IGES和STEP都是与CAD/CAM系统提供中性产品数据的公共资源和应用模型，它涉及到土建工程、机械、结构、电气、电子工程及船舶结构等领域，为了解决数据在不同的CAD/CAM系统之间进行数据传送的问题，定义了一套表示CAD/CAM系统中常用的几何和非几何数据格式以及相应的文件结构。

不同点：

IGES是1982年ANSI标准，而STEP是ISO/IEC JTCL下的SC4开发的ISO标准，它克服了IGES的一些缺点：

（1）不能精确地完整地转换数据，其原因是不同的CAD/CAM系统之间许多概念不一样，使得某些定义数据像表面定义数据会丢失

（2）不能转换属性信息

（3）层信息常丢失

（4）不能把两个零部件的信息放在一个文件中

（5）产生的数据量太大，以至许多CAD系统难以处理（无论是时间还是存储容量上都不适应）

（6）在转换数据的过程发生的错误很难确定，常常需要人工去处理IGES

文件，对此要花费大量的时间和精力。

而STEP克服了IGES中存在的问题，扩大了转换CAD/CAM系统中几何拓扑数据的范围,STEP即产品模型数据的公共资源和应用模型。STEP的产品模型数据是覆盖产品整个生命周期的应用而全面定义的产品所有数据元。在STEP中采用了形状特征信息模型进行各种产品模型定义数据的转换，强调建立能存入数据库中的一个产品模型的完整表示，而不只是它的图形或可视的表示

IGES采用了对实体单元进行数据描述的文件结构，而STEP采用了形状特征信息模型进行各种产品模型定义数据的转换的概念模式；

IGES的文件格式为目录入口、参数节、整体节、结束节和定义信息5个节；STEP的产品信息分为应用层、逻辑层和物理层3个层结构。

第三章：P163

1、你所用的图形软件是属于子程序库、专业语言和交互命令，还是这三种形式的混合形式，或是其他的形式。你认为你所用的图形软件的成功和不足之处是什么？有哪些改进意见？

（1）使用的图形软件采用C语言作为主语言，属于子程序库、专用语言和交互命令混合；

（2）成功之处：使用方便、便于扩充、便于用户加入用户自己编写的源程序或目标代码；不足之处是格式随所用主语言格式而定，修改源程序比较麻烦；

（3）实现与其他语言（非主语言）的兼容运行，交互式修改运行结果，而不需要人为地查看、修改源程序。

2、面向应用程序的接口通常有哪几种形式？你认为哪一种形式更方便应用和扩充应用功能？

面向应用程序的接口有子程序库、专用语言和交互命令三种形式；

交互任务是用户最关注的事，交互技术是完成交互任务的手段，故交互命令是用户接口中应用最普遍、效率最高的一种形式，对交互设备、交互任务、交互技术以及控制方式等的综合处理是完成交互命令和实现交互命令的依据。总的来说，交互命令更方便应用和扩充应用功能。

3、请列出你所用的交互系统中所涉及到的交互任务和交互技术，是否有本章书中没有提及的交互任务和交互技术？若有，能否对其进行分解，使之和本章书中介绍的交互任务和交互技术相匹配。

就目前有用过的交互任务和交互技术而言，均在本章范围内；

（1）交互任务：区域选择、文本输入、定路径和控制

（2）交互技术：选择技术、定位技术、定路径技术、文本技术、徒手画技术和拖动技术。

6、常见的交互任务有哪几种？你认为哪一种交互任务最难完成？

常见的交互任务有8种

定位、选择、文本、定向、定路径、定量、三维交互任务、组合交互任务

其中三维交互任务和组合交互任务最难完成，因为三维交互任务涉及定位、选择和旋转，用户难以区分屏幕上游标选择到对象的深度值和其他显示对象的深度值。

组合交互任务主要包含对话框、构造和动态控制三种，其实现前面几种的结合，是动态的，故其完成难度最大。

7、常见的交互技术有哪几种？你认为哪一种交互技术最容易使用？

交互技术指通过[计算机](http://www.itisedu.com/phrase/200603021438435.html)输入、输出设备，以有效的方式实现人与计算机对话的技术。它包括机器通过输出或显示设备给人提供大量有关信息及提示请示等，人通过输入设备给机器输入有关信息及提示请示等，人通过输入设备给机器输入有关信息，回答问题等。[人机交互](http://www.itisedu.com/phrase/200604221212315.html)技术是计算机用户界面设计中的重要内容之一。它与认知学、人机工程学、心理学等学科领域有密切的联系。

常见的交互技术有:选择技术、定位技术、定向技术、定路径技术、定量技术、文本技术、橡皮筋技术、徒手画技术、拖动技术；其中定量技术最容易使用。

11、交互式用户接口常见的工作方式有几种？你认为哪一种较实用？

用户接口工具箱使你能够为你的应用程序建立复杂的用户接口。用户接口可以包括窗口、对话、菜单条、附件（域、列表框，等等）、标记和公用对话（文件选择对话、信息对话，等等）。用户接口工具箱包括各种公用对话，例如选择一个文件或显示错误信息的对话。为了使用一个公用的对话，只需简单地实例化它，设置合适的参数，以及把它连接到你的应用程序中。

固定域输入输出方式

问答方式

表处理方式

命令语言

菜单方式

图形符号方式

12、请用菜单驱动方式、数据表格驱动方式和事件驱动方式完成同一个实际的交互任务。并比较它们之间的难易程度和工作量。

事件驱动的真实工作过程

数据表格驱动方式：

程序和表格关联，用表格（如EXCEL）的形式将变量传递到程序中进行一系列操作，或再用表格输出，不同于消息驱动的WINDOWS程序机制。使用表格驱动建立菜单，可以很方便很简单地管理自定义菜单，节约开发与维护成本。

《**太平洋保险终端系统**》交易界面实现了参数化的配置方式，系统使用屏幕定义文件和表格定义文件来定义交易屏幕，表格定义文件定义了表格的基本属性和表格的列，屏幕定义文件定义了屏幕的基本属性和屏幕上的栏位和表格的位置。屏幕和表格定义文件都需要引用数据字典里面的栏位定义。

界面驱动包括屏幕驱动和表格驱动，其功能包括设置栏位缺省值和缺省属性（是否可输入、是否必输入、是否绑定下拉菜单pklist），响应各种系统事件（屏幕前后事件、栏位前后事件、最后栏位后事件等），响应热键（ESC、DEL、PAGEUP、PAGEDOWN、CTRL\_XX等），动态改变屏幕属性（多页显示的翻页、拆分屏幕等），发起二段式交易等等。屏幕驱动和表格驱动是提供给开发人员的统一接口，开发人员通过实现该接口的方式开发功能模块。

事件驱动方式:  
1、启动应用程序，装载和显示窗体，产生Form\_Load和Form\_Show事件    
2、窗体或窗体上的控件接收事件，事件可由用户引发（例如键盘或鼠标操作），可由系统引发（例如定时器事件），也可由代码间接引发（例如当代码装载其他窗体时产生的Load事件）    
3、如果在相应的事件过程中存在代码，就执行代码    
4、应用程序等待下一次事件    
例：.获取和修改计算机名字的方法    
1.）插入一个新模块，在其中添加如下代码：    
  
′声明 GetComputerName   
Declare Function GetComputerName Lib〃kernel 32〃Alias〃   
GetComputerNameA〃(Byval lpBuffer As    
String,nSize As Long)As Long   
′声明 SetComputerName   
Declare Function SetComputerName Lib〃kernel 32〃Alias 〃   
SetComputerNameA〃(Byval lp ComputerName As String)As Long   
′定义一个获取计算机名字的函数   
Public Function GetCName (CName) As Boolean   
Dim sComputerName As String ’计算机的名字   
Dim lComputerName As Long    
’计算机名字的长度   
Dim lResult As Long    
’GetComputerName的返回值   
Dim RV As Boolean   
′GetCName返回值，若为TRUE则表示操作成功   
lComputerNameLen=256   
sComputerName=Space (lComputerNameLen)   
lResult=GetComputerName (sComputerName,lCompputerNameLen)   
If lResult 〈〉0 Then Cname=Left＄ (sComputerName,lComputerNameLen)   
RV=True   
Else RV=False   
End If   
GetCName=RV   
End Function   
′定义一个修改计算机名字的函数   
Public Function SetCName (CName ) As Boolean   
Dim lResult As Long   
Dim RV As Boolean   
lResult=SetComputerName (CName)   
If lResult 〈〉0 Then   
RV=True′修改成功   
Else RV=False   
End If   
SetCName=RV   
End Function   
  
2）.在窗体中添加一命令按钮Command1，双击该按钮并在其中添加如下代码：    
Sub Command1－Click ()   
DIM CN AS String   
x=GetCName (CN)   
Print 〃This Computer Name is :〃,CN   
CN=〃MYCOMPUTER〃   
x=SetCName (CN )   
Print 〃Now the Computer name is :〃,CN   
End Sub

1）对于完成同一个实际的任务来说，一般而言菜单驱动方式较数据表格驱动方式和事件驱动方式容易

2）相对于用户的工作量来说，菜单驱动方式较数据表格驱动方式和事件驱动方式的工作量大。

第四章：P215

**1、将中点画线算法推广以便能画出任意斜率的直线**

算法设计：

1. 输入直线的起点坐标P0(x0,y0)和终点坐标P1(x1,y1).
2. 定义直线当前点坐标x和y，定义中点偏差判别式d、直线斜率k、像素点颜色rgb
3. x= x0,y= y0计算d=0.5-k,k=( y1-y0)/(x1-x0), rgb=RGB=(0,0,255).
4. 绘制点(x,y)，判断d的符号，若d<0，则(x, y)更新为(x+1,y+1)，d更新为d+1-k，否则(x, y)更新为(x+1,y)，d更新为d-k.
5. 如果当前点x小于(x1，重复步骤(4)，否则结束。

程序主要代码：

MidPointLine(x0,y0,x1,y1,color)

{

int a,b,delta1,delta2,d,x,y;

a = y0 – y1;

b = x1 – x0;

d = 2\*a – b;

delta1 = 2 \* a;

delta2 = 2 \* (a+b);

x = x0;

y = y0;

if (a<b)

drawpixel(x, y, color);

else

drawpixel(y,x,color);

while (x > x1)

{

If (d<0)

{

x++;

y++;

d+ = delta2;

}

Else

{

X++;

D+=delta1;

}

Putpixel(x,y,color);

}

Else

While (x<x1)

{

If (d<0)

{

x--;

y++;

d-=delta3;

}

Else

{

x--;

d-=delta1;

}

Putpixel(x,y,color);

}

}

**2、采用整数Bresenham算法，为一台计算机编制直线扫描转换程序。从键盘敲入两端点坐标，就能在显示器屏幕上画出对应的直线。**

Void DrawLine(int color)

{

int x0,y0,x1,y1,color, I;

scanf( “%d, %d, %d, %d”, &x0, &y0, &x1, &y1);

dx=x1 – x0;

dy=y1 – y0;

e = -dx;

x = x0;

y = y0;

for ( i=0; i<=dx; i++)

{

putpixel(x, y, color);

x=x+1;

e=e+2\*dy;

if (e>=0)

{

y = y + 1;

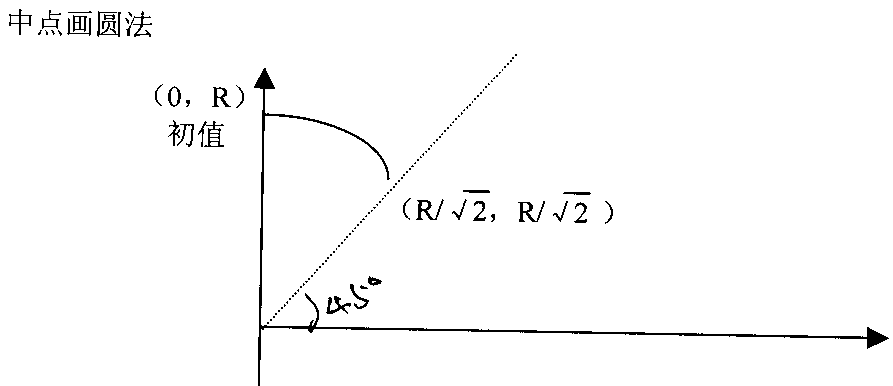
e = e – 2 \* dy;

}

}

}

**4、试编写按逆时针方向生成第二个8分圆的中点算法**



算法设计：

1. 输入圆的半径
2. 定义圆当前点坐标x和y、中点偏差判别式d、像素点颜色rgb
3. 计算d=1.25-R,x=0,y=R, rgb=RGB=(0,0,255).
4. 绘制点(x, y)，及其在八分圆中的另外7个对称点‘
5. 判断d的符号，若d<0，则(x, y)更新为(x+1,y)，d更新为d+2x+3，否则(x, y)更新为(x+1,y-1)，d更新为d+2(x-y)+5.
6. 当x小于等于y，重复步骤(4)和（5），否则结束。

MidpointCircle(r,color)

int r, color;

{

float x,y;

float d;

x=0;

y=r;

d=1.25 –r/1.414;

drawpixel(x, y, color);

while (x<y)

{

if (d<0)

{

d+=2\*x+3;

x++;

}

else

{

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

x++;

y--;

}

drawpixel(x,y,color);

}

}

**5、假设圆的圆心不在原点，试编写算法对整个圆进行扫描转换**

算法设计：

1. 输入圆的半径r, 圆心坐标为(xc,yc)
2. 定义圆当前点坐标x和y、中点偏差判别式d、像素点颜色rgb
3. 计算d=1.25-R,x=0,y=R, rgb=RGB=(0,0,255).
4. 绘制点(x+xc, y+yc)，及其在八分圆中的另外7个对称点‘
5. 判断d的符号，若d<0，则(x, y)更新为(x+1,y)，d更新为d+2x+3，否则(x, y)更新为(x+1,y-1)，d更新为d+2(x-y)+5.
6. 当x小于等于y，重复步骤(4)和（5），否则结束。

(1)用bresenham画圆法，设圆心坐标为(xc,yc)

bresenham\_circle(r,color)

int r,color;

{

int x,y,delta,delta1,delta2,direction;

x=xc;

y=yc+r;

delta=2\*(1-r);

while (y>=0)

{

drawpixel(x,y,color);

if (delta<0)

{

delta1=2\*(delta+y-b)-1;

if (delta1<=0)

direction=1;

else

direction=2;

}

else if (delta>0)

{

delta2=2\*(delta-x+xc)-1;

if (delta2<=0)

direction=2;

else

direction=3;

}

else

direction=2;

switch (direction)

{

case 1:x++;

delta+=2\*(x-xc)+1;

break;

case 2:x++;

y++;

delta+=2\*(x-a-y+yc+1);

break;

case 3:y--;

delta+=-2\*(y-yc)+1;

break;

}

}

}

(2) 采用中点画圆算法

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

MidpointCircle(r,color)

int r, color;

{

float x,y;

float d;

x=0;

y=r;

d=1.25 –r/1.414;

putpixel(xc+x, yc+y, color);

while (x<y)

{

if (d<0)

{

d+=2\*x+3;

x++;

}

else

{

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

x++;

y--;

}

putpixel(xc+x,yc+y,color);

}

}

putpixel(x,y,color);

putpixel(xc+yc-y,yc+xc-x,color);

putpixel(xc+yc-y,yc+xc-x,color);

putpixel(x, yc+yc-y,color);

putpixel(xc+xc-x,yc+yc-y,color);

putpixel(xc-yc+y,yc-xc+x,color);

putpixel(xc-yc+y,yc+xc-x,color);

putpixel(xc+xc-x,y,color);

}

void main()

{

int gdriver=DETECT, gmode,xc,yc,r;

initgraph(&gdriver, &gmode, “”);

printf(“Please enter the xc:”);

scanf(“%d”, &xc);

printf(“Please enter the yc:”);

scanf(“%d”, &yc);

printf(“Please enter the R:”);

scanf(“%d”, &r);

cleardevice();

MidpointCircle(xc,yc, r,RED);

getch();

closegraph();

}

**6、试编写可以对一段任意圆弧进行扫描转换的算法**

将360度的区域分成8个部分

3 2

4 1

5 8

6 7

编写可以对一段任意圆弧进行扫描转换的算法的关键在于，对这段圆弧的起点和终点分别判定是否在同一区域

如果起点和终点在同一区域，调用中点画圆算法，但要根据实际情况对参数进行修正；

如果起点和终点不在同一区域，则要根据实际情况对圆弧段进行分割，分割的原则是将每一段的起点和终点放在同一区域，然后分别调用中点画圆算法画圆弧，同样在画的过程中，要根据实际情况对参数进行修正及算法进行修正；

设圆弧的起点为(x1,y1),终点为(x2,y2)，半径为r

如图

A（x1,y1）

D(x0-r,y0) C (x0,y0)

B(x2,y2)

将整个圆弧分为两段，弧AC和弧CB，分别进行扫描转换，转换过程中利用中点画圆方法进行，代码如下：

midpoint(x1,y1,x2,y2,r,color,k)

{

int x,y;

float d;

x=x1;

y=y1;

d=(x1+1)^2+(y1-0.5)^2-r^2;

putpixel(x,y,color);

while (x<=x2)

{

if (d<0)

{

d+=2\*x+3;

x++;

}

else

{

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

x++;

y=y+k;

}

}

putpixel(x,y,color);

main()

{

scanf(“%d”,&n);//分割的圆弧数

for (i=1;i<=n;i++)

{

scanf(“%d,%d,%d,%d,%d”,&x1,&y1,&x2,&y2,,&k); //要求x1<x2

midpoint(x1,y1,x2,y2,r,color,k);

**7、设计一个多边形区域填充算法，使其边界像素具有一个值，而内部的像素具有另一个值。**

算法设计：

1. 使用画线语句绘制多边形
2. 计算窗口客户区的水平边界最大值MaxX和垂直边界最大值MaxY
3. 调用系统调色板，设置颜色值FillColor为调色板上取得的颜色，CBackColor为白色。
4. 对于每一条边，y从ymin开始，执行下面的循环。
5. x从扫描线和边的交点处开始到窗口客户区右边界，先获得(x, y)位置的像素颜色，如果是填充色，则置成背景色，否则所有填充色填充。执行x=x+1/k，计算下一个x起点值。
6. 如果y=ymin，则扫描结束，否则y++，转（5）。

主要代码：

int MaxX,MaxY;

Void GetMaxX() //求屏幕最大x值

{

CRect rect;

GetClientRect(rect);

MaxX=rect.riht;

}

Void GetMaxY() //求屏幕最大y值

{

CRect rect;

GetClientRect(rect);

MaxX=rect.bottom;

}

Void Draw() //填充多边形函数

{

COLORREF CBackColor=RGB(255,255,255);//白色

CClientDC dc(this);

int m,n,ymin,ymax;

double x,y,k;

for (int i=0; i<=6; i++)

{

m=i,n=i+1;

if (7==n) n=0;

k=(double (Point [m].x- Point [n].x)/ (Point [m].y- Point [n].y);//计算1/k;

if ((Point [m].y< Point [n].y) //得到每条边y的最大和y最小值

{

ymin= Point [m].y;

ymax= Point [n].y;

x=Point [m].x; //得到x|ymin

}

else

{

ymin= Point [n].y;

ymax= Point [m].y;

x=Point [n].x;

}

For (y=ymin;y<ymax;y++)

{

For(int j=ROUND(x);j<MaxX;j++)//对每一条扫描线与边的交点的右侧像素循环

{

If(dc.GetPixel(j, ROUND(y)==FillColor)//如果像素的颜色是填充色

{

dc.SetPixel(j, ROUND(y),CBackColor);//改为背景色

}

else

{

dc.SetPixel(j, ROUND(y),FillColor);//改为填充色

}

}

x+=k; //计算下一个x起点值

}

}

}

DrawPolygon() //绘制多边形函数

{

CClientDC dc(this);

int m,n;

for (int j=0;j<=6;j++)

{

M=j;n=j+1;

If (7==n) n=0;

dc.MoveTo(Point[m]);

dc.LineTo(Point[n]);

}

}

**10、试设计一个生成具有宽度的直线条的算法，使得在直线条连接处不出现图示的缺口**

对于第一种方案：

c

d

e

a b f

可以把整个线条分成两个部分，

对于左半部分：

先计算出线条的四个顶点，a,b,c,d

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

同样，对于右半部分，依法处理

对于第二种方案：

c

e

d f

g

a b h

可以把整个线条分成三个部分，

对于左半部分：

先计算出线条的四个顶点，a, b, c, d

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

对于中间部分：

先计算出线条的四个顶点，c, d， e，f

再用半径为某个值如R圆弧分别把c, e和 d, f连接起来，

最后调用区域填充算法把所得的圆环段c, e, d, f进行填色

同样，对于右半部分，依法处理

先计算出线条的四个顶点，e, f, g, h

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

12、为26个英文大写字母设计5X7的字符掩膜矩阵。

***13、编写一程序实现线段裁剪的中点分割算法***

算法设计：

（1）输入直线段的两端点坐标：P0(x0,y0)，P1(x1,y1)绘制坐标为(wxl, wyt),(wxr,wyb)的窗口

（2）P0点的编码为RC0, P1点的编码为RC1。

（3）若RC0|RC1=0，对直线段应“取”，转步骤（6）；否则若RC0|RC1≠0，对直线段应“弃”，转步骤（6）。

（4）如果直线段有一个端点在窗口内，则计算该直线的中点坐标P[x=(x0+x1)/2,y=(y0+y1)/2]，并计算其编码RC。如果中点P和P0在规定的误差范围内（例如10-6）不重合，则判断中点是否也在窗口内，如果在，PP0 “取”之，否则，PP0 “弃”之。将中点P赋给点P1，转步骤（6）。

（5）如果直线段的两个端点都不在窗口内，则必定与窗口相交，求该直线中点坐标P[x=(x0+x1)/2,y=(y0+y1)/2]和其编码RC。当中点P在窗口外时，如果RC0&RC1=0，则PP1在窗口外，“弃”之；否则PP0在窗口外，“弃”之，直到中点P在窗口内。对于中点在窗口内的直线段PP1和PP0，重复步骤（4）。

（6）输出裁减后的线段。

主要代码：

Void MidClip(double P0x, double P0y, double P1x, double P1y,BOOL flag)

{

Double x,y;

Unsigned int RCT0,RCT1;

RCT0=EnCode(P0x,P0y);

RCT1=EnCode(P1x,P1y);

x=( P0x+ P1x)/2;

y=( P0y+ P1y)/2; RCT=EnCode(x,y);

while(abs(x-P0x)>1e-6|| abs(y-P0y)>1e-6)

{

if(RCT==0)//中点也在窗口内，则P=P0

{

P0x=x;

P0y=y;

RCT0= RCT;

}

else //否则舍弃P1点

{

P1x=x;

P1y=y;

RCT1= RCT;

}

x=( P0x+ P1x)/2;

y=( P0y+ P1y)/2; RCT=EnCode(x,y);

}

if(flag==true)

{

Pointx[1]=x;

Pointy[1]=y;

}

else

{

Pointx[0]=x;

Pointy[0]=y;

}

}

***14 编写一程序实现逐次多边形裁剪算法***

算法设计：

（1）输入第一个顶点坐标：F(x0,y0)

第二个顶点坐标：S(x1,y1)

(2) 当顶点输入完毕，转（7）

（3）输入顶点P坐标：P(x2,y2)

（4）SP与裁剪线相交吗？是，求SP与裁剪线的交点I（x,y）,并输出I坐标：I(x,y)

（5）P位于可见一侧吗？是，输出顶点P坐标：P(x2,y2)

（6）将顶点P坐标：P(x2,y2)=》顶点S坐标：S(x1,y1)，转（2）

（7）将顶点F坐标：F(x0,y0)=》顶点P坐标：S(x2,y2)，形成闭合，

（8） SP与裁剪线相交吗？是，求SP与裁剪线的交点I（x,y）,并输出I坐标：I(x,y)

（9）P位于可见一侧吗？是，输出顶点P坐标：P(x2,y2)

（10）结束

程序代码：

/\* Sutherland-Hodgman 算法 \*/

#define LEN sizeof(struct node)

#include <math.h>

#include "display.h"

struct node {

int dx,dy;

struct node \*next;

};

struct node \*creat()

{ struct node \*h,\*q,\*r;

int p[8][2]={100,120,160,50,180,100,200,80,240,160,210,220,170,160,140,190};

int i;

setcolor(12);

for (i=0;i<7;i++) line(p[i][0],p[i][1],p[i+1][0],p[i+1][1]);

line(p[0][0],p[0][1],p[7][0],p[7][1]);

rectangle(120,200,230,70);

h=NULL;

for (i=0;i<8;i++)

{ q=(struct node \*)malloc(LEN);

q->dx=p[i][0]; q->dy=p[i][1];

if (h==NULL) h=q;

else r->next=q;

r=q;

}

r->next=NULL;

return(h);

}

struct node \*builx(h,x)

struct node \*h;

int x;

{int s[2],j[2];

struct node \*hh,\*p,\*r,\*q;

int max,min;

p=h; hh=NULL;

s[0]=p->dx; s[1]=p->dy;

p=p->next;

while (p!=NULL)

{ j[0]=x;

j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]);

max=s[0]; min=p->dx;

if (s[0]<p->dx) { max=p->dx; min=s[0]; }

if ((j[0]>=min)&&(j[0]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0]; q->dy=j[1];

if (hh==NULL) hh=q;

else r->next=q;

r=q;

}

if (p->dx>=x)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx; q->dy=p->dy;

if (hh==NULL) hh=q;

else r->next=q;

r=q;

}

s[0]=p->dx; s[1]=p->dy;

p=p->next;

}

p=h;

j[0]=x; j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]);

max=s[0]; min=p->dx;

if (s[0]<p->dx) { max=p->dx; min=s[0]; }

if ((j[0]>=min)&&(j[0]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0]; q->dy=j[1];

if (hh==NULL) hh=q;

else r->next=q;

r=q;

}

if (p->dx>=x)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx; q->dy=p->dy;

if (hh==NULL) hh=q;

else r->next=q;

r=q;

}

r->next=NULL;

return(hh);

}

struct node \*builxx(h,x)

struct node \*h;

int x;

{int s[2],j[2];

struct node \*hh,\*p,\*r,\*q;

int max,min;

p=h;

hh=NULL;

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

while (p!=NULL)

{

j[0]=x;

j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]+0.1);

max=s[0];

min=p->dx;

if (s[0]<p->dx) { max=p->dx;

min=s[0];

}

if ((j[0]>=min)&&(j[0]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dx<=x)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

}

p=h;

j[0]=x;

j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]+0.1);

max=s[0];

min=p->dx;

if (s[0]<p->dx) { max=p->dx;

min=s[0];

}

if ((j[0]>=min)&&(j[0]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dx<=x)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

r->next=NULL;

return(hh);

}

struct node \*buily(h,y)

struct node \*h;

int y;

{int s[2],j[2];

struct node \*hh,\*p,\*r,\*q;

int max,min;

p=h;

hh=NULL;

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

while (p!=NULL)

{

j[1]=y;

j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

max=s[1];

min=p->dy;

if (s[1]<p->dy) { max=p->dy;

min=s[1];

}

if ((j[1]>=min)&&(j[1]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dy>=y)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

}

p=h;

j[1]=y;

j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

max=s[1];

min=p->dy;

if (s[1]<p->dy) { max=p->dy;

min=s[1];

}

if ((j[1]>=min)&&(j[1]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dy>=y)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

r->next=NULL;

return(hh);

}

struct node \*builyy(h,y)

struct node \*h;

int y;

{int s[2],j[2];

struct node \*hh,\*p,\*r,\*q;

int max,min;

p=h;

hh=NULL;

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

while (p!=NULL)

{

j[1]=y;

j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

max=s[1];

min=p->dy;

if (s[1]<p->dy) { max=p->dy;

min=s[1];

}

if ((j[1]>=min)&&(j[1]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dy<=y)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

s[0]=p->dx;

s[1]=p->dy;

p=p->next;

}

p=h;

j[1]=y;

j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

max=s[1];

min=p->dy;

if (s[1]<p->dy) { max=p->dy;

min=s[1];

}

if ((j[1]>=min)&&(j[1]<=max))

{ q=(struct node \*)malloc(LEN);

q->dx=j[0];

q->dy=j[1];

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

if (p->dy<=y)

{ q=(struct node \*)malloc(LEN);

q->dx=p->dx;

q->dy=p->dy;

if (hh==NULL)

hh=q;

else r->next=q;

r=q;

}

r->next=NULL;

return(hh);

}

main()

{

int max,min;

struct node \*head,\*r,\*q;

int i;

int s[2];

Initialize();

head=creat();

q=head;

while (q->next!=NULL)

{ putpixel(q->dx,q->dy,14);

q=q->next;

}

putpixel(q->dx,q->dy,14); /\*左边界进行裁减\*/

q=builx(head,120);

head=q;

while (q->next!=NULL)

{ putpixel(q->dx,q->dy,15);

q=q->next;

}

putpixel(q->dx,q->dy,15); /\*上边界进行裁减\*/

q=buily(head,70);

head=q;

while (q->next!=NULL)

{ putpixel(q->dx,q->dy,2);

q=q->next;

}

putpixel(q->dx,q->dy,2); /\*右边界进行裁减\*/

q=builxx(head,230);

head=q;

while (q->next!=NULL)

{ putpixel(q->dx,q->dy,1);

q=q->next;

}

putpixel(q->dx,q->dy,1); /\*下边界进行裁减\*/

q=builyy(head,200);

head=q;

s[0]=q->dx;

s[1]=q->dy;

q=q->next;

setcolor(14);

outtextxy(100,100,"abc");

while (q!=NULL)

{ line(s[0],s[1],q->dx,q->dy);

s[0]=q->dx;

s[1]=q->dy;

q=q->next;

getch();

}

q=head;

outtextxy(200,200,"def ");

line(s[0],s[1],q->dx,q->dy);

getch();

}

第六章 曲线和曲面

**3、参照Hermite三次曲线的几何形式，试用B[P0 P1 P0u P1u P0uu P1uu]T , 推导相应五次曲线的调和函数和系数矩阵M。**

解：设Hermite五次曲线的几何形式为：

P(t)=a5t5 + a4t4 + a3t3 + a2t2 + a1t + a0 其中 t∈[0,1]

按题意，已知曲线两端点的坐标值P0 P1

曲线两端点的一阶导数值P0u P1u

曲线两端点的二阶导数值P0uu P1uu

则求出系数a5,a4,a3,a2,a1,a0

则P(t)就可确定；

由于P(t)= a5t5 + a4t4 + a3t3 + a2t2 + a1t + a0 其中 t∈[0,1]

P’(t)=5a5t4 + 4a4t3 + 3a3t2 + 2a2t + a1

P”(t)=20a5t3+12a4t2+6a3t+2a2

P0=P(0)=a0

P1=P(1)=a5+a4+a3+a2+a1+a0

P0’=P’(0)=a1

P1’=P’(1)=5a5+4a4+3a3+2a2+a1

P0”=P”(0)=2a2

P1”=P”(1)=20a5+12a4+6a3+2a2

所以 a0 = P(0)

a1 =P’(0)

a2 =P”(0)/2

a3 = 10P(1)- 10P(0) - 4P’(1) - 6P’(0) + P”(1)/2 - 3P”(0)/2

a4 =-15P(1)+ 15P(0) + 7P’(1) + 8P’(0) - P”(1) - 3P”(0)/2

a5 = 6P(1)- 6P(0) - 3P’(1) - 3P’(0) - P”(0)/2 + P”(1)/2

=>

P(t)=[ -6P(0) + 6P(1) - 3P’(0) - 3P’(1) - P”(0)/2 + P”(1)/2] t5

+[+15P(0) - 15P(1) + 8P’(0) + 7P’(1) + 3P”(0)/2 ] t4

+[-10P(0) + 10P(1) - 6P’(0) - 4P’(1) - 3P”(0)/2 + P”(1)/2] t3

+ [ P”(0)/2] t2

+ [P’(0)] t

+P(0)

整理得：

P(t) = (-6t5 + 15t4 - 10t3 + 1) P(0) + (6t5-15t4+10t3) P(1)

+ (-3t5 + 8t4 -6t3 + t) P’(0) + (-3t5 +7t4-4t3) P’(1)

+ (-t5/2+ 3t4/2-3t3/2+t2/2) P”(0) + (t5/2-t4+t3/2) P”(1)

故调和函数为：

F(0)= -6t5 + 15t4 - 10t3 + 1

F(1)= 6t5 - 15t4 + 10t3

F(2)= -3t5 + 8t4 - 6t3 + t

F(3)= -3t5 + 7t4- 4t3

F(4)= -t5/2 + 3t4/2 -3t3/2 + t2/2

F(5)= t5/2 - t4 + t3/2

系数矩阵为：

- 6 6 -3 -3 -1/2 1/2

15 -15 8 7 3/2 -1

-10 10 -6 -4 -3/2 1/2

0 0 0 0 1/2 0

0 0 1 0 0 0

1 0 0 0 0 0

**9．试求两段三次Hermite曲线达C1和G1连续的条件**

解：两段三次Hermite曲线分别为：

Q1(t1)=a3 t13 + a2 t12+ a1 t1+ a0 t1∈[0 1]

Q2(t2)=b3 t23 + b2 t22+ b1 t2+ b0 t2∈[0 1]

(1)依据G1连续充要条件为：

Q1(1)和Q2(0)在P点处重合，

且其在P点处的切矢量方向相同，大小不等

即 Q1(1)= Q2(0)， Q1’(1)≠ Q2’(0) ,Q1”(1)= Q2”(0)

而 Q1(1)= a3 + a2 + a1 + a0

Q2(0)= b0

Q1’(t1)=3a3 t12 + 2a2 t1+ a1

Q2’(t2)=3b3 t22 + 2b2 t2+ b1

Q1’(1)=3a3 + 2a2+ a1

Q2’(0)= b1

Q1”(t1)=6a3 t1 + 2a2

Q2”(t2)=6b3 t2 + 2b2

Q1”(1)=6a3 + 2a2

Q2”(0)= 2b2

=> 两段三次Hermite曲线：

Q1(t1)=a3 t13 + a2 t12+ a1 t1+ a0 t1∈[0 1]

Q2(t2)=b3 t23 + b2 t22+ b1 t2+ b0 t2∈[0 1]

要达到G1连续，其系数必须满足下列关系式：

a3 + a2 + a1 + a0 = b0

3a3 + 2a2 + a1 ≠ b1

6a3 + 2a2 =2 b2

(2)依据C1连续充要条件为：

Q1(1)和Q2(0)在P点处重合，

且其在P点处的切矢量方向相同，大小相等

即 Q1(1)= Q2(0)， Q1’(1)= Q2’(0) ,Q1”(1)= Q2”(0)

而 Q1(1)= a3 + a2 + a1 + a0

Q2(0)= b0

Q1’(t1)=3a3 t12 + 2a2 t1+ a1

Q2’(t2)=3b3 t22 + 2b2 t2+ b1

Q1’(1)=3a3 + 2a2+ a1

Q2’(0)= b1

Q1”(t1)=6a3 t1 + 2a2

Q2”(t2)=6b3 t2 + 2b2

Q1”(1)=6a3 + 2a2

Q2”(0)= 2b2

=> 两段三次Hermite曲线：

Q1(t1)=a3 t13 + a2 t12+ a1 t1+ a0 t1∈[0 1]

Q2(t2)=b3 t23 + b2 t22+ b1 t2+ b0 t2∈[0 1]

要达到C1连续，其系数必须满足下列关系式：

a3 + a2 + a1 + a0 = b0

3a3 + 2a2 + a1 = b1

6a3 + 2a2 =2 b2

**10.给定四点P1（0,0,0）,P2(1,1,1),P3(2,-1,-1),P4(3,0,0),用其作为特征多边形来构造一条三次Bezier曲线，并计算参数为0，1/3，2/3，1的值。**

解：三次Bezier曲线的一般式为：

P(t)=(1-t)3P1 +3t(1-t)2P2+ 3t2(1-t)P3+t3P4 t∈[0 1]

**其矩阵表达式为**

-1 3 -3 1 P1

P(t)=[t3 t2 t 1] 3 -6 3 0 P2

-3 3 0 0 P3

1 0 0 0 P4

-1 3 -3 1 0 0 0

**=** [t3 t2 t 1] 3 -6 3 0 1 1 1

-3 3 0 0 2 -1 -1

1 0 0 0 3 0 0

**0**  6 6

**=>** P(t)=[t3 t2 t 1] 0 -9 -9

3 3 3

0 0 0

**然后分别令t=0, 1/3, 2/3, 1 计算上述式子即可**

**当t＝0时**

**P(0)= 0 0 0 1 0 6 6**

**0 -9 -9**

**3 3 3**

**0 0 0**

**= 0 0 0**

**当t＝1/3时**

**P(1/3)= 1/33 1/32  1/3 1 0 6 6**

**0 -9 -9**

**3 3 3**

**0 0 0**

**= 1 2/9 2/9**

**当t＝2/3时**

**P(2/3)= (2/3)3 (2/3)2 2/3 1 0 6 6**

**0 -9 -9**

**3 3 3**

**0 0 0**

**= 2 -2/9 2/9**

**当t＝1时**

**P(0)= 1 1 1 1 0 6 6**

**0 -9 -9**

**3 3 3**

**0 0 0**

**= 3 0 0**

**11．已知由P1（0，0，0）,P2（2，2，-2）,P3（2，-1，-1）,P4（3，0，0）,Q1（4，0，0）,Q2（6，-2，1）,Q3（8，-3，2）,Q4（10，0，1）确定的两段三次Bezier曲线，试求其在P4(Q1)处达到C1连续的条件**

**解：**设两段连续的三次Bezier曲线分别为：

P(t), Q(t) t∈[0 1]

则 **P(t1)=(1-t1)3P1+3t1(1-t1)2P2+3t12(1-t1)P3+t13P4**

t1∈[0 1]

**Q(t2)=(1-t2)3Q1+3t2(1-t2)2Q2+3t22(1-t2)Q3+t23Q4**

t2∈[0 1]

将P1、P2、P3、P4的分量分别代入P(t)得到相应的分量

Px(t)= **(1-t)3\*0 + 3t(1-t)2\*2 + 3t2(1-t)\*2 + t3\*4**

**= 4t3 – 6t2 + 6t**

Py(t)= **(1-t)3\*0 + 3t(1-t)2\*2 + 3t2(1-t)\*(-1) + t3\*0**

**= 9t3 – 15t2 + 6t**

Pz(t)= **(1-t)3\*0 + 3t(1-t)2\*(-2) + 3t2(1-t)\*(-1) + t3\*0**

**= -3t3 + 9t2 - 6t**

即三次Bezier曲线的矩阵式为：

P(t)= [t3 t2 t 1] 4 9 -3 0

-6 -15 9 0

6 6 -6 0

0 0 0 0

将Q1、Q2、Q3、Q4的分量分别代入Q(t)得到相应的分量

Qx(t)= **(1-t)3\*4 + 3t(1-t)2\*6 + 3t2(1-t)\*8 + t3\*10**

**= 6t + 4**

Qy(t)= **(1-t)3\*0 + 3t(1-t)2\*(-2) + 3t2(1-t)\*(-3) + t3\*0**

**= 3t3 + 3t2 - 6t**

Qz(t)= **(1-t)3\*0 + 3t(1-t)2\*1 + 3t2(1-t)\*2 + t3\*1**

**= -2t3 + 3t**

即三次Bezier曲线的矩阵式为：

Q(t)= [t3 t2 t 1] 0 3 -2 0

0 3 0 0

6 -6 3 0

4 0 0 4

P(t)和Q(t)在P4(Q1)处达到C1连续的条件是：

P(1)和Q(0) 在P4(Q1)处重合，且其在在P4(Q1)处的切矢量方向相同，大小相等

即：

P（t=1） = Q（t=0）

P’（t=1） = Q’（t=0）

**12、把上题定义的二段三次Bezier曲线经过一次分割后，试求它们相应特征多边形的顶点坐标序列。**

解：顶点坐标序列如图：

P1

P2 P11

P3 P21 P12

P4 P31 P22 P13

Q2 Q11

Q3 Q21 Q12

Q4 Q31 Q22 Q13

其中Pi r (t1)=（1-t）Pi r-1 (t1)+ t Pi+1 r-1 (t1) r=1,2,3; i=1,2,3

**17、已知P00=[0.25,0], P10=[0.75,0],P01=[0.75,0.9],P11=[0.25,0.8]四点，试用它们构造一张双线性曲面，并用程序输出该双线性曲面。**

解：一次双线性曲面参数方程为：

P00 P01 1-v

Q(u, v)=[1-u, u] P10 P11 v

[0.25,0] [0.75,0.9] 1-v

=[1-u, u] [0.75,0] [0.25,0.8] v

即：

Q(u, v)=（1-u）(1-v) P00+u (1-v) P10+（1-u）v P01+uv P11

=（1-u）(1-v) [0.25,0]+u (1-v) [0.75,0]+（1-u）v [0.75,0.9]+u v[0.25,0.8]

即：

X(u, v)= 0.25 (1-u）(1-v) +0.75u (1-v) +0.75（1-u）v+0.25u v

Y(u, v)= 0 (1-u）(1-v) + 0 u (1-v) + 0.9（1-u）v+ 0.8u v 0≤u≤1; 0≤v≤1

它其实是一个马鞍面（双曲抛物面）上的一块曲面片。

P01 P11

P00 P10

**7.6 习题**

**2．试证明下述几何变换的矩阵运算具有互换性：**

**（1）两个连续的旋转变换；（2）两个连续的平移变换；**

**（3）两个连续的变比例变换；（4）当比例系数相等时的旋转和比例变换；**

（1）证明：设第一次的旋转变换为：

cosθ1 sinθ1 0

T1= - sinθ1 cosθ1 0

0 0 1

第二次的旋转变换为：

Cosθ2 sinθ2 0

T2= - sinθ2 cosθ2 0

0 0 1

则因为

T1\*T2 = cosθ1 sinθ1 0 cosθ2 sinθ2 0

- sinθ1 cosθ1 0 - sinθ2 cosθ2 0

0 0 1 0 0 1

= cosθ1 cosθ2+sinθ1 sinθ2 cosθ1 sinθ2+ sinθ1 cosθ2 0

- sinθ1 cosθ2- cosθ1 sinθ2 -sinθ1 sinθ1+ cosθ1 cosθ2 0

0 0 1

Cos（θ1+θ2） sin（θ1+θ2） 0

= - sin（θ1+θ2） cos（θ1+θ2） 0

0 0 1

cosθ2 sinθ2 0 cosθ1 sinθ1 0

T2\*T1 = - sinθ2 cosθ2 0 - sinθ1 cosθ1 0

0 0 1 0 0 1

cosθ1 cosθ2+ sinθ1 sinθ2 cosθ1 sinθ2+ sinθ1 cosθ2 0

= - sinθ2cosθ1- cosθ2 sinθ1 -sinθ1 sinθ1+ cosθ1 cosθ2 0

0 0 1

Cos（θ1+θ2） sin（θ1+θ2） 0

= - sin（θ1+θ2） cos（θ1+θ2） 0

0 0 1

即T1\*T2= T2\*T1, **两个连续的旋转变换具有互换性**

（2）证明：设第一次的平移变换为：

1 0 0

T1= 0 1 0

Tx1 Ty1 1

第二次的平移变换为：

1 0 0

T2= 0 1 0

Tx2 Ty2 1

则因为

T1\*T2 = 1 0 0 1 0 0

0 1 0 0 1 0

Tx1 Ty1 1 Tx2 Ty2 1

1 0 0

= 0 1 0

Tx1+Tx2 Ty1+Ty2 1

而

T2\*T1 = 1 0 0 1 0 0

0 1 0 0 1 0

Tx2 Ty2 1 Tx1 Ty1 1

1 0 0

= 0 1 0

Tx1+Tx2 Ty1+Ty2 1

即T1\*T2= T2\*T1, **两个连续的平移变换具有互换性**

（3）证明：设第一次的变比例变换为：

Sx1 0 0

T1= 0 Sy1 0

0 0 1

第二次的变比例变换为：

Sx2 0 0

T2 = 0 Sy2 0

0 0 1

则因为

T1\*T2 = Sx1 0 0 Sx2 0 0

0 Sy1 0 0 Sy2 0

0 0 1 0 0 1

Sx1\*Sx2 0 0

= 0 Sy1\*Sy2 0

0 0 1

而

T2\*T1 = Sx2 0 0 Sx1 0 0

0 Sy2 0 0 Sy1 0

0 0 1 0 0 1

Sx1\*Sx2 0 0

= 0 Sy1\*Sy2 0

0 0 1

即T1\*T2= T2\*T1, **两个连续的变比例变换具有互换性**

（4）证明：设第一次为比例系数相等时的比例变换：

S 0 0

T1= 0 S 0

0 0 1

第二次的为旋转变换：

cosθ sinθ 0

T2= - sinθ cosθ 0

0 0 1

则因为

T1\*T2 = S 0 0 cosθ sinθ 0

0 S 0 - sinθ cosθ 0

0 0 1 0 0 1

S cosθ S sinθ 0

= - S sinθ S cosθ2 0

0 0 1

而

T2\*T1 = cosθ sinθ 0 S 0 0

- sinθ cosθ 0 0 S 0

0 0 1 0 0 1

S cosθ S sinθ 0

= -S sinθ S cosθ 0

0 0 1

即T1\*T2= T2\*T1, **“当比例系数相等时的旋转和比例“变换具有互换性**

**3、证明二维点相对x轴作对称，紧跟着相对y=-x直线作对称变换完全等价于该点相对坐标原点作旋转变换。**

证明：

(1) 点相对x轴作对称的变换矩阵

1 0 0

T1= 0 -1 0

0 0 1

(2) 相对于y=-x直线作对称变换矩阵

0 -1 0

T2= -1 0 0

0 0 1

1 0 0 0 -1 0 0 -1 0

因为 T1\*T2= 0 -1 0 \* -1 0 0 = 1 0 0

0 0 1 0 0 1 0 0 1

cos(-90º) sin(-90 º) 0

= - sin(-90 º) cos(-90º) 0

0 0 1

即该点相对坐标原点作顺时针方向转90 º的旋转变换

1. **证明**

**1-t2 2t**

**1+ t 2 1+t2**

**T= 完全表示一个旋转变换。**

**-2t 1-t2**

**1+t2 1+t2**

证明：令t=tg(θ/2)

则：（1-t2）/(1+ t 2)= cosθ

（2t）/(1+ t 2)= sinθ

即

cosθ sinθ

T=

- sinθ cosθ

将T扩充为一个三行齐次坐标的变换矩阵为：

cosθ sinθ 0

T= - sinθ cosθ 0

0 0 1

该矩阵表示为一个旋转变换

**5、例：三角形ABC各顶点坐标为A（3，0）B（4，2）C（6，0），其绕原点逆时针旋转90°，再向X方向平移2，Y方向平移-1。**

解：因为：θ=90°

变换矩阵为

**COS90° SIN90° 0 0 1 0**

**TR= - SIN90° COS90° 0 = -1 0 0**

**2 -1 1 2 -1 -1**

则

**A 3 0 1 0 1 0 2 2 1 A‘**

**B 4 2 1 -1 0 0 = 0 3 1 B‘**

**C 6 0 1 2 -1 1 2 5 1 C‘**

如果先进行平移变换，再进行旋转变换，

1 0 0 **COS90° SIN90° 0 0 1 0**

Tr= 0 1 0 **- SIN90° COS90° 0 = -1 0 0**

**2 -1 1 0 0 1 1 2 1**

**则**

**A 3 0 1 0 1 0 1 5 1 A‘**

**B 4 2 1 -1 0 0 = -1 6 1 B‘**

**C 6 0 1 1 2 1 1 8 1 C‘**

**结论：变换顺序不同，结果也不同**