# 基本知识点

1、在图形文件系统中，点、线、圆等图形元素通常都用其几何特征参数来描述，在图形系统中，图形处 理运算的精度不取决于显示器的分辨率，在彩色图形显示器中，使用 RGB 颜色模型。计算机图形学以计 算几何为理论基础。

2、深度缓存算法并不需要开辟一个与图像大小相等的深度缓存数组，深度缓存算法能并行实现，深度缓 存算法中没有对多边形进行排序。

3、计算机图形处理中，除了应用到各种算法外，还经常会处理大量的图形方面的数据，因而必须应用到 数据库技术，图形数据库设计的子库层次是一个简单的、具有普遍存储规则的许多物体的集合，图形数据 库的设计一般有物体和子库两个层次。

4、投影线从视点出发，主灭点最多有 3 个，任何一束不平行于投影面的平行线的透视投影将汇成一点。 在平面几何投影中，若投影中心移到距离投影面无穷远处，则成为平行投影。 5、实体模型和曲面造型是CAD 系统中常用的主要造型方法，曲面造型是用参数曲面描述来表示一个复杂 的物体，从描述复杂性和形状灵活性考虑，最常用的参数曲面是3 次有理多项式的曲面，在曲线和曲面定 义时，使用的基函数应有两个重要性质：凸包性和仿射不变性。

6、简单光反射模型，又称为 Phong 模型，它模拟物体表面对光的反射作用，简单光反射模型主要考虑物 体表面对直射光照的反射作用，在简单光反射模型中，对物体间的光反射作用，只用一个环境光变量做近 似处理。

7、定义了物体的边界也就唯一的定义了物体的几何形状边界，物体的边界上的面是有界的，而且，面的 边界应是闭合的，物体的边界上的边可以是曲线，但在两端之间不允许曲线自相交。

8、透视投影的投影线从视点出发，主灭点最多有 3 个，任何一束不平行于投影面的平行线的透视投影将 汇成一点。

9、图形数据按照目的不同一般可以分为图形的表示数据和图形的显示数据。

10、双线性法向插值法（Phong Shading）的优点是高光域准确。

11、画圆弧的算法有角度 DDA 法、逐点比较法、终点判断法、Bresenham画圆法四种。

12、Z 缓冲器消隐算法是最简单的消除隐藏面算法之一。

13、若要对某点进行比例、旋转变换，首先需要将坐标原点平移至该点，在新的坐标系下做比例或旋转变换，然后再将原点平移回去。

14、在种子填充算法中所提到的八向连通区域算法同时可填充四向连通区。

15、多边形被两条扫描线分割成许多梯形，梯形的底边在扫描线上，腰在多边形的边上，并且相间排列， 多边形与某扫描线相交得到偶数个交点，这些交点间构成的线段分别在多边形内、外，并且相间排列，边 的连贯性告诉我们，多边形的某条边与当前扫描线相交时，很可能与下一条扫描线相交。

16、透视投影又可分为一点透视、二点透视、三点透视，斜投影又可分为斜等测、斜二测，正视图又可分 为主视图、侧视图、俯视图。

17、Bezier 曲线不一定通过其特征多边形的各个顶点，Bezier 曲线两端点处的切线方向必须与起特征折线 集（多边形）的相应两端线段走向一致，Bezier 曲线可用其特征多边形来定义。

18、扫描线算法对每个象素只访问一次，主要缺点是对各种表的维持和排序的耗费较大 ，边填充算法基 本思想是对于每一条扫描线与多边形的交点，将其右方象素取补，边填充算法较适合于帧缓冲存储器的图 形系统。

19、深度缓冲器算法最简单常用的面向应用的用户接口形式：子程序库、专用语言和交互命令。图形用户 界面的基本元素有窗口、图标、菜单、指点装置。在计算机图形学中，被裁剪的对象可以是线段、多边形 和字符三种形式。

20、扫描仪最重要的参数是光学精度和扫描精度。

21、图形处理标准：GKS，GKS-3D，PHIGS，CGM，CGI，IGES，STEP

22、坐标系：建模坐标系，用户坐标系（直角坐标系，仿射坐标系，圆柱坐标系，球坐标线，极坐标系），观察坐标系，规格化的设备坐标系，设备坐标系

23、输入模式：请求方式，取样方式，事件方式，组合方式

1、计算机图形学：用计算机建立、存储、处理某个对象的模型，并根据模型产生该对象图形输出的有关 理论、方法与技术，称为计算机图形学。

2、计算机图形标准：计算机图形标准是指图形系统及其相关应用程序中各界面之间进行数据传送和通信 的接口标准。

5、输入设备：键盘、鼠标、光笔、触摸屏、操纵杆、跟踪球和空间球、数据手套、图像扫描、数字化仪、声频视频输入系统

显示设备：CRT显示器、液晶显示器以及等离子显示器

6、生成直线常用的算法主要有DDA法、中点画线法、直线的Bresenham算法。

7、掌握：画圆、椭圆的算法（Bresenham）

8、填充算法：重点是种子填充算法（代码）

9、字符的图形表示分为点阵字符和矢量字符。

10、走样：在光栅显示器上绘制非水平且非垂直的直线或多边形边界时，或多或少会呈现锯齿状。这是由 于直线或多边形边界在光栅显示器的对应图形都是由一系列相同亮度的离散像素构成的。这种用离散量表 示连续量引起的失真，称为走样（aliasing）。

11、反走样：用于减少和消除用离散量表示连续量引起的失真效果的技术，称为反走样。

12、二维图形的基本变换主要有平移、比例，对称，旋转，错切等。

13、投影分为正投影和斜投影。

14、灭点：对于透视投影，若一组平行线平行于投影平面时，它们的透视投影仍然保持平行，而不平行于投影面的平行线的投影会聚到一个点，这个点就是灭点。

15刷新式CRT图形显示器按扫描方式分为随机扫描和光栅扫描两种。

16、图形消隐：计算机为了反映真实的图形，把隐藏的部分从图中消除。

17、几何变换：几何变换的基本方法是把变换矩阵作为一个算子，作用到图形一系列顶点的位置矢量，从 而得到这些顶点在几何变换后的新的顶点序列，连接新的顶点序列即可得到变换后的图形。

18、计算几何：计算几何研究几何模型和数据处理的学科，讨论几何形体的计算机表示、分析和综合，研 究如何方便灵活、有效地建立几何形体的数学模型以及在计算机中更好地存贮和管理这些模型数据。

19、裁剪：识别图形在指定区域内和区域外的部分的过程称为裁剪算法，简称裁剪。

20、透视投影：空间任意一点的透视投影是投影中心与空间点构成的投影线与投影平面的交点。

21、透视投影按照灭点个数可分为 一点透视 、两点透视、三点透视三种。

22、投影变换：把三维物体变为二维图形表示的变换称为投影变换。

23、窗口：世界坐标的范围是无限大的。为了使规格化设备坐标上所显示的世界坐标系中的物体有一个合 适的范围与大小，必须首先对世界坐标系指定显示范围，它通常是一个矩形，这个矩形被称为窗口。

24、视区：在规格化设备坐标系上也要指定一个矩形区域与窗口对应，显示窗口里的内容，这个矩形被称 为视区。

25、坐标系统：为了描述、分析、度量几何物体的大小、形状、位置、方向以及相互之间的各种关系使用 的参考框架叫做坐标系统。

26、刷新：荧光的亮度随着时间按指数衰减，整个画面必须在每一秒内重复显示许多次，人们才能看到一 个稳定而不闪烁的图形，这叫屏幕的刷新。

27、用户坐标系：用户坐标系用户为处理自已的图形时所采用的坐标系，单位由用户自己决定。

28、规范化设备坐标系：将各个设备坐标系中的数据化为统一的数据范围从而得到的设备坐标系。

29、规格化变换：图形软件根据窗口与视区的一一对应关系，自动实现从世界坐标到规格化设备坐标的转 换，这种从窗口到视区的变换，称为规格化变换。

30、屏幕坐标系统：屏幕坐标系统也称设备坐标系统，它主要用于某一特殊的计算机图形显示设备(如光栅 显示器)的表面的点的定义，在多数情况下，对于每一个具体的显示设备，都有一个单独的坐标系统，在定 义了成像窗口的情况下，可进一步在屏幕坐标系统中定义称为视图区的有界区域，视图区中的成像即为实 际所观察到的图形对象。

31、观察坐标系：观察坐标系通常是以视点的位置为原点，通过用户指定的一个向上的观察向量来定义整 个坐标系统，缺省为左手坐标系，观察坐标系主要用于从观察者的角度对整个世界坐标系内的对象进行重 新定位和描述，从而简化几何物体在投影面的成像的数学推导和计算。

32、简要指出下列术语之间的主要不同之处：

图形和图象：前者是指矢量表示的图，后者是指用点阵表示的图。

随机扫描与光栅扫描：前者是电子束扫描路径随图形不同而不同，后者是电子束扫描路径固定不变。

窗口与视口：前者是在世界（用户）坐标系中定义的，后者是设备坐标系中定义的。

数组与矩阵：前者是计算机语言中的术语，后者是数学中的术语。

逼近与插值：前者是所求曲线大致通过数据点，后者是所求曲线通过每一个数据点。

33、图形交互技术有哪些？

定位技术、定向技术、定路经技术、定量技术文本技术、橡皮筋技术、拖动技术

1、计算机图形学：用计算机建立、存储、处理某个对象的模型，并根据模型产生该对象图形输出的有关 理论、方法与技术，称为计算机图形学。

2、计算机图形标准：计算机图形标准是指图形系统及其相关应用程序中各界面之间进行数据传送和通信 的接口标准。

3、图形消隐：计算机为了反映真实的图形，把隐藏的部分从图中消除。

4、几何变换：几何变换的基本方法是把变换矩阵作为一个算子，作用到图形一系列顶点的位置矢量，从 而得到这些顶点在几何变换后的新的顶点序列，连接新的顶点序列即可得到变换后的图形。

5、计算几何：计算几何研究几何模型和数据处理的学科，讨论几何形体的计算机表示、分析和综合，研究如何方便灵活、有效地建立几何形体的数学模型以及在计算机中更好地存贮和管理这些模型数据。 6、裁剪：识别图形在指定区域内和区域外的部分的过程称为裁剪算法，简称裁剪。

7、透视投影：空间任意一点的透视投影是投影中心与空间点构成的投影线与投影平面的交点。

8、投影变换：把三维物体变为二维图形表示的变换称为投影变换。

9、走样：在光栅显示器上绘制非水平且非垂直的直线或多边形边界时，或多或少会呈现锯齿状。这是由 于直线或多边形边界在光栅显示器的对应图形都是由一系列相同亮度的离散像素构成的。这种用离散量表 示连续量引起的失真，称为走样（aliasing）。

11、窗口：世界坐标的范围是无限大的。为了使规格化设备坐标上所显示的世界坐标系中的物体有一个合 适的范围与大小，必须首先对世界坐标系指定显示范围，它通常是一个矩形，这个矩形被称为窗口。

12、视区：在规格化设备坐标系上也要指定一个矩形区域与窗口对应，显示窗口里的内容，这个矩形被称 为视区。

13、坐标系统：为了描述、分析、度量几何物体的大小、形状、位置、方向以及相互之间的各种关系使用 的参考框架叫做坐标系统。

14、刷新：荧光的亮度随着时间按指数衰减，整个画面必须在每一秒内重复显示许多次，人们才能看到一 个稳定而不闪烁的图形，这叫屏幕的刷新。

15、用户坐标系：用户坐标系用户为处理自已的图形时所采用的坐标系，单位由用户自己决定。

16、规范化设备坐标系：将各个设备坐标系中的数据化为统一的数据范围从而得到的设备坐标系。

17、规格化变换：图形软件根据窗口与视区的一一对应关系，自动实现从世界坐标到规格化设备坐标的转 换，这种从窗口到视区的变换，称为规格化变换。

18、屏幕坐标系统：屏幕坐标系统也称设备坐标系统，它主要用于某一特殊的计算机图形显示设备(如光栅 显示器)的表面的点的定义，在多数情况下，对于每一个具体的显示设备，都有一个单独的坐标系统，在定 义了成像窗口的情况下，可进一步在屏幕坐标系统中定义称为视图区的有界区域，视图区中的成像即为实际所观察到的图形对象。

19、观察坐标系：观察坐标系通常是以视点的位置为原点，通过用户指定的一个向上的观察向量来定义整 个坐标系统，缺省为左手坐标系，观察坐标系主要用于从观察者的角度对整个世界坐标系内的对象进行重 新定位和描述，从而简化几何物体在投影面的成像的数学推导和计算。

3、简要指出下列术语之间的主要不同之处：

图形和图象：前者是指矢量表示的图，后者是指用点阵表示的图。

随机扫描与光栅扫描：前者是电子束扫描路径随图形不同而不同，后者是电子束扫描路径固定不变。

窗口与视口：前者是在世界（用户）坐标系中定义的，后者是设备坐标系中定义的。

数组与矩阵：前者是计算机语言中的术语，后者是数学中的术语。

逼近与插值：前者是所求曲线大致通过数据点，后者是所求曲线通过每一个数据点。

4、图形交互技术有哪些？

回显，引力域、草拟技术、旋转、形变、网格、约束、橡皮筋技术、拖动技术

定位技术、定向技术、定路经技术、定量技术、文本技术

5、DDA算法生成直线的基本原理是什么？

是从直线的微分方程中生成直线。由于直线的一阶导数是连续的，而且△ X和△ Y是成比例的，因此可以通过当前位置( Xi, Yi) 分别加上两个小增量

ε\*△ X和ε\*△ y (ε为无穷小的正数)来求出下一个点( Xi+1,Yi+1)的x，y的坐标，

即有： Xi+1= Xi +ε\*△ X；

Yi+1= Yi +ε\*△ y 。

6、计算机图形学研究的内容

如何在计算机中表示图形，以及利用计算机进行图形的计算，处理和显示的相关原理和算法

计算机动画 地理信息系统 人机交互 真实感图形显示 虚拟现实、科学计算可视化 并行图形处理

二维变换矩阵划分成四个子矩阵：左上角（比例，旋转，对称，错切）左下角（平移）右上角（投影），右下角（整体比例）

字符裁剪按照精度可分为：字符精度、串精度、笔画/像素精度

**-------------------------------------------------------------------------------**

# 一、选择题

1、下列有关平面几何投影的叙述语句中，正确的论述为（ C ）

A）透视投影变换中，一组平行线投影在与之平行的投影面上，会产生灭点

B）透视投影与平行投影相比，视觉效果更有真实感，而且能真实地反映物体的精确的尺寸和形状

C）在平面几何投影中，若投影中心移到距离投影面无穷远处，则成为平行投影

D）在三维空间中的物体进行透视投影变换，可能产生三个或者更多的生灭点。

2、透视投影中主灭点最多可以有几个? ( D ) A）0； B）1； C）2； D）

3、触摸屏是( C )设备。 A）输入 B）输出 C）输入输出 D）既不是输入也不是输出

4、下面关于反走样的论述哪个是错误的？( D )

A）提高分辨率； B）把象素当作平面区域进行采样；

C）采用锥形滤波器进行加权区域采样；D）增强图象的显示亮度 ；

5、在下列叙述语句中，不正确的论述为（ C ）

A. 在图形文件系统中，点、线、圆等图形元素通常都用其几何特征参数来描述；

B. 在图形系统中，图形处理运算的精度不取决于显示器的分辨率；

C.在光栅扫描图形显示器中，所有图形都按矢量直接描绘显示，不存在任何处理；

D. 在彩色图形显示器中，使用RGB 颜色模型。

6、在多边形面片数量很大时；消隐算法最快的应该是（ C ）

A Z-Buffer B 扫描线 C 画家算法 D 不确定

7、计算机图形学与计算几何之间的关系是 ( B )。

A）学术上的同义词

B）计算机图形学以计算几何为理论基础

C）计算几何是计算机图形学的前身

D）．两门毫不相干的学科

8、分辨率为1024\*1024 的显示器，其位平面数为24，则帧缓存的字节数应为（ A ）

A 3MB B 2MB C 1MB D 512KB

9、下列有关物体的几何表示法的叙述语句中，正确的论述为（ ）

A） 在计算机图形学中，通常所谓"物体"是三维欧氏空间点的集合；

B） 一组三维欧氏空间点的集合都可看成一个（组）"物体"；

C） 单个孤立的点是"物体"; D） 一根直线段或单张曲面都是"物体"。

10、图形软件系统提供给用户三种基本的输入方式，不包含的选项是（ B ）

A）请求方式； B）采样方式； C）事件方式； D）随机方式。

11、计算机图形显示器一般使用什么颜色模型？

( A ) 4 A）RGB； B）CMY； C）HSV ； D）HLS

12、下述用数值微分法画斜率的绝对值小于1 的直线的C 语言子程序中哪一行有错？( D ) Void drawLineWithDDA (int x1, int y1, int x2, int y2, int color) {

A） int x, y;

B） float k = (float)(y2-y1)/(x2-x1);

C） for(x=x1,y=y1;x<=x2;x++)

D）{ drawPixel(x,y,color); y+=k; } }

13、在下述三维齐次变换矩阵中，平移线性变换对应的矩阵元素的最大非零个数是（ C ） a b c l d e f m

A）3; B）6; g h I n C）7; D）8 p q r s

14、多边形填充时，下述哪个论述是错误的？（ C ）

A. 多边形被两条扫描线分割成许多梯形，梯形的底边在扫描线上，腰在多边形的边上，并且相间 排列；

B. 多边形与某扫描线相交得到偶数个交点，这些交点间构成的线段分别在多边形内、外，并且相 间排列；

C. 在判断点是否在多边形内时，一般通过在多边形外找一点，然后根据该线段与多边形的交点数 目为偶数即可认为在多边形内部，若为奇数则在多边形外部，而且不需考虑任何特殊情况；

D. 边的连贯性告诉我们，多边形的某条边与当前扫描线相交时，很可能与下一条扫描线相交；

15、下面关于NURBS 的论述，哪个是错误的？（ B ）

A. 可通过控制顶点和权因子来改变形状； B. 仅具有仿射不变性，但不具有透射不变性；

C. 非有理

B 样条、有理及非有理Bezier 曲线、曲面是NURBS 的特例；

D. 可表示标准解析形状和自由曲线、曲面；

16、在光亮度插值算法中，下列论述哪个是错误的？（ D ）

A. Gouraud 明暗模型计算中，多边形与扫描平面相交区段上每一采样点的光亮度值是由扫描平面 与多边形边界交点的光亮度插值得到的；

B. Phong 明暗处理模型中，采用了双线性插值和构造法向量函数的方法模拟高光；

C. Gouraud 明暗模型和Phong 明暗处理模型主要是为了处理由多个平面片近似表示曲面物体的绘 制问题；

D. Phong 明暗模型处理的物体表面光亮度呈现不连续跃变；

18、在三维几何造型方法中，局部操作能力比较弱的方法是A

A 体素造型 B 八叉树造型 C B-rey 造型 D 特征造型

1、灰度等级为256极，分辨率为1024\*1024的显示器，至少需要的帧缓存容量为B

A）512KB B）1MB C）2MB D）3MB

2、哪一个2不是国际标准化组织（ISO）批准的图形标准？D

A）GKS B）PHIGS C）CGM D）DXF

3、光笔是一种（ ）设备。A

A）输入 B）输出 C）输入输出 D）非输入也非输出

4、CRT显示器的带宽与下列哪个因素有关？A

A）分辨率 B）电压 C）电流 D）点距

5、显示器每秒钟刷新的次数成为B

A）行频 B）场频 C）带宽 D）点距

6、下列哪种设备不是图形输入设备：D

A）鼠标 B）键盘 C）数字化仪 D）绘图仪

7、下列属于图形输入设备的是（）A

A）数字化仪 B）打印机 C）绘图仪 D）显示器

8、扫描仪属于图形（）设备A

A）显示 B）输入 C）显示与输入 D）以上都不对

9、多边形填充时，下述哪个论述是错误的？C

A.多边形被两条扫描线分割成许多梯形，梯形的底边在扫描线上，腰在多边形的边上，并且相间排列；

B.多边形与某扫描线相交得到偶数个交点，这些交点间构成的线段分别在多边形内、外，并且相间排列；

C. 在判断点是否在多边形内时，一般通过在多边形外找一点，然后根据该线段与多边形的交点数目为偶数即可认为在多边形内部，若为奇数则在多边形外部，而且不需考虑任何特殊情况；

D.边的连贯性告诉我们，多边形的某条边与当前扫描线相交时，很可能与下一条扫描线相交；

10、下面哪一项操作不能达到反走样的目的？D

A.提高分辨率； B. 把象素当作平面区域进行采样；

C.采用锥形滤波器进行加权区域采样； D.增强图象的显示亮度 ；

11、在用户坐标系下定义的区域为B

A 窗口区和视图区 B 窗口区 C 视图区 D 以上都不是

12、在三维齐次变换矩阵中，平移线性变换对应的矩阵元素的最大非零个数是C

A）3 B）6 C）7 D）8

13、在三维齐次变换矩阵中，均匀的整体放大变换对应的矩阵元素的非零非1个数是C

A）1 B）2 C）3 D）4

14、在多边形的逐边裁剪法中,对于某条多边形的边(方向为从端点S到端点P)与某条裁剪线(窗口的某一边)的比较结果共有以下四种情况,分别需输出一些顶点.请问哪种情况下输出的顶点是错误的?A

A) S和P均在可见的一侧,则输出S和P

B) S和P均在不可见的一侧,则输出0个顶点

C) S在可见一侧,P在不可见一侧,则输出线段SP与裁剪线的交点

D) S在不可见的一侧,P在可见的一侧,则输出线段SP与裁剪线的交点和P

15透视投影中主灭点最多可以有几个?D

A）0 B）1 C）2 D）3

16光笔是一种（ ）设备。A

A）输入 B）输出 C）输入输出 D）非输入也非输出

17显示器每秒钟刷新的次数成为B

行频 B）场频 C）带宽 D）点距

18下列说法正确的是：A

平移变换只改变图形的位置，不改变图形的大小和形状

B)平移变换不改变图形的位置，只改变图形的大小和形状

C)平移变换能改变图形的位置，也能改变图形的大小和形状

D)平移变换不改变图形的位置，也不改变图形的大小和形状

**31.计算机绘图设备一般使用什么颜色模型？( B )**

A）RGB； B）CMY； C）HSV ； D）HLS A

**-------------------------------------------------------------------------------**

# 二、多选题

1． 关于光栅扫描式图形显示器，具有这样的特点：（B ，C ）

A. 桢缓存和光栅显示器均是数字设备 ；

B. 需要足够的位面和桢缓存才能反映图形的颜色和灰度 ；

C. 对于彩色光栅显示器的R.G,B 三原色需要三个位面的桢缓存和三个个电子枪 ；

D. 对原色配置颜色查找颜色查找表的目的是为了提高显示的速度；

2． 关于计算机图形标准化的论述，哪个是正确的（A，C ，D ） ；

A. CGM和CGI 是面向图形设备的接口标准 ；

B. GKS、IGES、STEP 均是ISO 标准 ；

C. IGES 和STEP 是数据模型和文件格式的标准；

D. PHIGS 具有模块化的功能结构；

3． 用户接口形式包括：（ A，B，C ）； A. 子程序库； B. 专用语言； C. 交互命令； D. 交互标准 ；

4． 下列关于Bezier 曲线的性质，哪个是正确的？（A，C，D ）

A. 在起点和终点处的切线方向和控制多边形第一条边和最后一条边的方向一致；

B. 在端点处的R 阶导数，仅与R 个相邻个控制顶点有关；

C. 曲线及其控制多边形在起点处有什么几何性质，在终点处也有什么性质；

D. 对于平面曲线而言，其与某直线的交点个数不多于该直线与控制多边形的交点个数；

5． 关于NURBS 中的权因子W，下面的那些论述是正确的？（ B，C ）

A. 随着W的增/减，曲线则远离/靠近相应的控制顶点；

B. 相应于某给定的参数的NURBS 曲线上的点，在W不同时将扫描出一条直线段；

C. 若NURBS 曲线上的点趋向相应的控制顶点，则权因子趋向正无穷；

D. 若控制顶点相同，所有的权因子分别扩大10 倍，所得的曲线将比原曲线更逼近控制顶点；

6． 以下关于图形变换的论述那些是正确的？（A,C，D ）

A. 平移变换不改变图形大小和形状，只改变图形位置 ；

B. 错切变换虽然可引起图形角度的改变，但不会发生图形畸变；

C. 拓扑关系不变的几何变换不改变图形的连接关系和平行关系；

D. 旋转变换后各图形部分间的线性关系和角度关系不变，变换后直线的长度不变；

7． 在几何造型中，下列论述中那些是正确的？（ C D ）

A. 在形体定义中，允许存在孤立的点；

B. 面是形体上一个有限、非零的区域，一般由一个外环和若干个内环组成，但也可以没有任何环；

C. 环是有序、有向边组成的面的封闭边界，环中的边不能相交，相邻的两条边可以共享一个端点；

D. 形体上任意一点的足够小的邻域在拓扑上应该是一个等价的封闭圆；

8． 分数维造型方法中，下列论述中哪个是正确的？（A，B，D ）

A. 二维空间中的一个分数维曲线的维数在1-2 之间；

B. 三维空间中的一个分数维曲线的维数在1-3 之间；

C. 三维空间中的一个分数维曲面的维数在1-3 之间；

D. 三维空间中的一个分数维曲面的维数在2-3 之间；

9． 在各种消隐算法中，下列那些论述是正确的？ （A，B，C，D ）

A. 画家算法的基本思想是先将屏幕赋值为背景色，然后在把物体各个面按其到视点距离远近排 序；

B. Z 缓冲算法不仅需要桢缓冲区存放象素的亮度值，还需要一个 Z 缓冲区存放每个象素的深度 值；

C. 扫描线算法首先按扫描行顺序处理一桢画面，在由视点和扫描线所决定的扫描平面上解决消 隐问题；

D. 区域采样算法是利用图形的区域连贯性在连续的区域上确定可见面及其颜色和亮度；

10. 在光照明模型中，下列论述那些是正确的？ （A，B，C，D）

A. Lambert 漫反射模型模拟理想漫射表面的光亮度，没有考虑模型表面的镜面反射效果；

B. Phong 模型将表面漫反射光亮度和镜面反射光亮度看作为是对光源入射光的直接反射，且相互 独立；

C. Blinn 和Cook-Torrance 模型是对Phong 模型的改进，将镜面反射光是入射光的函数因素考虑进 来，因而产生的光照效果更加逼真；

D. 整体光照模型同时模拟了光源引起的漫反射分量、镜面反射分量，以及环境反射分量；

11． 在计算机图形的标准化论述中，下列是正确的结论有：AB A GKS、PHIGS 都是国际标准； B CGM、CGI 都是国际标准； C IGES、DXF 都是国际标准； D STEP 不是国际标准。 12． 在计算机图形关于Modeling 的物体的描述中，下列是不正确的结论有：ABD

A 一根直线是物体；

B 一个空间的点是物体；

C 一个立方体是物体；

D 三维欧氏空间点的集合是物体。

13． 凸包计算的主要用途在于：BC

A 多边形裁剪； B 区域填充； C 消隐； D 上述三种中的一个。 7

14．在下列叙述语句中，正确的论述为（ ABD ）

A: 在图形文件系统中，点、线、圆等图形元素通常都用其几何特征参数来描述；

B: 在图形系统中，图形处理运算的精度不取决于显示器的分辨率；

C: 在光栅扫描图形显示器中，所有图形都按矢量直接描绘显示，不存在任何处理；

D: 在彩色图形显示器中，使用RGB 颜色模型。

15．使用下列二维图形变换矩阵： 将产生变换的结果为（D ）

A: 图形放大2 倍；

B: 图形放大2 倍，同时沿X、Y 坐标轴方向各移动1 个绘图单位；

C: 沿X 坐标轴方向各移动2 个绘图单位；

D: 沿X 坐标轴方向放大2 倍，同时沿X、Y 坐标轴方向各平移1 个绘图单位。

16. 下列有关Bezier 曲线性质的叙述语句中，错误的结论为（BD ）

A: Bezier 曲线可用其特征折线集（多边形）来定义；

B: Bezier 曲线必须通过其特征折线集（多边形）的各个顶点；

C: Bezier 曲线两端点处的切线方向必须与起特征折线集（多边形）的相应两端线段走向一致； D: n 次Bezier 曲线，在端点处的r 阶导数，只与r 个相邻点有关。

17. 下列有关物体的几何表示法的叙述语句中，正确的论述为（AC ） A: 在计算机图形学中，通常所谓"物体"是三维欧氏空间点的集合； B: 一组三维欧氏空间点的集合都可看成一个（组）"物体"； C: 单个孤立的点不是"物体"; D: 一根直线段或单张曲面都是"物体"。

18. 下列有关边界表示法的叙述语句中，错误的论述为（C ）

A: 定义了物体的边界也就唯一的定义了物体的几何形状边界；

B: 物体的边界上的面是有界的，而且，面的边界应是闭合的；

C: 物体的边界上的面是有向的，面的法向总是指向物体的内部；

D: 物体的边界上的边可以是曲线，但在两端之间不允许曲线自相交。

19. 下列有关平面几何投影的叙述语句中，正确的论述为（ AC ）

A: 在平面几何投影中，若投影中心移到距离投影面无穷远处，则成为平行投影；

B: 透视投影与平行投影相比，视觉效果更有真实感，而且能真实地反映物体的精确的尺寸和形 状；

C: 透视投影变换中，一组平行线投影在与之平行的投影面上，不产生灭点；

D: 在三维空间中的物体进行透视投影变换，可能产生三个或者更多的生灭点。

20. 下列有关简单光反射模型的描述语句中，错误的论述为（ B ）

A: 简单光反射模型，又称为Phong 模型，它模拟物体表面对光的反射作用；

B: 在简单光反射模型中，假定光源是点光源，而且，仅仅关注物体表面对光的镜面反射作用；

C: 简单光反射模型主要考虑物体表面对直射光照的反射作用；

D: 在简单光反射模型中，对物体间的光反射作用，只用一个环境光变量做近似处理。

1、关于光栅扫描式图形显示器，具有这样的特点：BC

A.桢缓存和光栅显示器均是数字设备 ；

B. 需要足够的位面和桢缓存才能反映图形的颜色和灰度 ；

C.对于彩色光栅显示器的R.G,B三原色需要三个位面的桢缓存和三个个电子枪 ；

D.对原色配置颜色查找颜色查找表的目的是为了提高显示的速度；

2、在计算机图形的标准化论述中，下列是正确的结论有：AB

A GKS、PHIGS都是国际标准；

B CGM、CGI都是国际标准；

C IGES、DXF都是国际标准；

D STEP不是国际标准。

3、关于计算机图形标准化的论述，哪个是正确的ACD

A.CGM和CGI是面向图形设备的接口标准 ；

B.GKS、IGES、STEP均是ISO标准 ；

C.IGES和STEP是数据模型和文件格式的标准；

D.PHIGS具有模块化的功能结构；

4、在下列叙述语句中，正确的论述为ABD

A在图形文件系统中，点、线、圆等图形元素通常都用其几何特征参数来描述；

B在图形系统中，图形处理运算的精度不取决于显示器的分辨率；

C在光栅扫描图形显示器中，所有图形都按矢量直接描绘显示，不存在任何处理；

D在彩色图形显示器中，使用RGB颜色模型。

5、以下关于图形变换的论述那些是正确的？ACD

平移变换不改变图形大小和形状，只改变图形位置 ；

B.错切变换虽然可引起图形角度的改变，但不会发生图形畸变；

C.拓扑关系不变的几何变换不改变图形的连接关系和平行关系；

D.旋转变换后各图形部分间的线性关系和角度关系不变，变换后直线的长度不变；

# 三、填空题

1、按照工作原理分，目前显示器主要有CRT显示器、\_液晶显示器\_以及等离子显示器几种。

2、生成直线常用的算法主要有DDA法、＿中点画线法＿＿、直线的Bresenham算法。

3、二维图形的基本变换主要有平移、\_\_\_比例\_\_\_，对称，旋转，错切等。

4、在线段AB的区域编码裁剪算法中，如A、B两点的编码均为 零 （填汉字），则该线段位于窗口内；如A、B两点的码按位与不为零，则该线段在窗口外。

5、透视投影按照灭点个数可分为 一点透视 、两点透视、三点透视三种。

6、平面几何投影可以分为两大类，即＿ 正平行投影＿ 和斜平行投影。

7、刷新式CRT图形显示器按扫描方式分为随机扫描和 光栅扫描 两种。

8、常用的用户接口形式：子程序库，专用语言，交互命令

9、二维图形的基本变换主要有平移、比例，对称，旋转，错切等。

10、 生成直线的四点要求是 生成直线要直,直线终止点要准,直线粗细要均,速度要快

11、 可以用基本的交互技术形成三种组合交互任务：对话框，构造，操作。

12、 图形的基本要素一般有 点,线,面,体 .

13、处理图形时常涉及的几种坐标系有：用户坐标系、设备坐标系、建模坐标系、观察坐标系

1.平行几何投影可以分为两大类：正平行投影和斜平行投影

2. 生成直线的四点要求是 直线垂直，端点准确，亮度、色泽粗细均匀，划线速度尽可能快。

3. 将多边形外部一点A 与某一点B 用线段连接，若此线段与多边形边界相交的次数为 偶数次，则点B 在多边形外部，若此线段与多边形边界相交的次数为奇数次，则点B 在多边形内部。

4. 字符的图形表示可以分为点阵字符和矢量字符两种形式。

5. 处理图形时常涉及的几种坐标系有用户坐标系、设备坐标系、建模坐标系、观察坐标系。

6. 可以用基本的交互技术形成三种组合交互任务：对话框、构造、操作。

7. 平面几何投影根据投影线角度可以分为平行和透视。

8. 计算机图形系统中常用的用户接口形式有子程序库、专用语言和交互命令三种形式。

9.深度缓冲器算法最简单常用的面向应用的用户接口形式子程序库、专用语言和交互命令

10.图形用户界面的基本元素有窗口、图标、菜单、指点装置。在计算机图形学中被裁剪的对象可以是线段、多边形 和字符三种形式。

11.刷新式CRT图形显示器按扫描方式分为随机扫描和光栅扫描两种

12.生成直线常用的算法主要有DDA法、中点画线法、直线的Bresenham算法。

13.二维图形的基本变换主要有平移、比例，对称，旋转，错切等

14.在线段AB的区域编码裁剪算法中，如A、B两点的编码均为 零 （填汉字），则该线段位于窗口内；如A、B两点的码按位与不为零，则该线段在窗口外。

15.按照工作原理分，目前显示器主要有CRT显示器、液晶显示器以及等离子显示器几种。

16.透视投影按照灭点个数可分为一点透视、两点透视、三点透视三种。斜投影又可分为斜等测、斜二测，正视图又可分为主视图、侧视图、俯视图。

17.可以用基本的交互技术形成三种组合交互任务：对话框，构造，操作。 .

18.图形的基本要素一般有： 点，线ˍ，面，体

19.扫描仪最重要的参数是光学精度和扫描精度。

20.图形数据按照目的不同一般可以分为图形的表示数据和图形的显示数据。

21.双线性法向插值法（Phong Shading）的优点是高光域准确。

22.画圆弧的算法有角度 DDA 法、逐点比较法、终点判断法、Bresenham画圆法四种。

23.Z 缓冲器消隐算法是最简单的消除隐藏面算法之一。

24.曲线的数学表示形式通常有显式、隐含和参数三种

25.投影变换中，透视投影的投影中心到投影面之间的距离是有限的，平行投影的投影中心到投影面之间的距离是无限的。

# 四、判断题

T 1、一个逻辑输入设备可以对应多个物理输入设备。

F 2、光栅扫描式图形显示器可看作是点阵单元发生器，可直接从单元阵列中的一个可编地址的象素画一条直线到另一个可编地址的象素。

T 3、计算机图形学标准通常是指数据文件格式标准和子程序界面标准。

T 4、阴极射线管的技术指标主要是分辨率和显示速度。

F 5、液晶显示器LCD的优点是：价低、重量轻、尺寸小且消耗功率低。

F 6、计算机图形生成的基本单位是线段。

T 7、直线的生成算法是二维图形生成技术的基础。

T 8、在种子填充算法中所提到的八向连通区域算法同时可填充四向连通区。

F 9、边填充算法中是将扫描线与多边形交点左方的所有象素取补。

F 10、齐次坐标提供了坐标系变换的有效方法，但仍然无法表示无穷远的点。

1.计算机图形学标准通常是指数据文件格式标准和子程序界面标准-T

2.计算机图形生成的基本单位是线段F

3.在种子填充算法中所提到的八向连通区域算法同时可填充四向连通区T

4.齐次坐标系不能表达图形中的无穷远点F

5.一个逻辑输入设备可以对应多个物理输入设备。T

6.光栅扫描式图形显示器可看作是点阵单元发生器，可直接从单元阵列中的一个可编地址的象素画一条直线到另一个可编地址的象素。F

7.齐次坐标提供了坐标系变换的有效方法，但仍然无法表示无穷远的点。F

8.阴极射线管的技术指标主要是分辨率和显示速度。T

9.液晶显示器LCD的优点是：价低、重量轻、尺寸小且消耗功率低。F

10.边填充算法中是将扫描线与多边形交点左方的所有象素取补。F

11.直线的生成算法是二维图形生成技术的基础。T

12.欧拉公式 v – e + f = 2 也适用于三维形体中的相关信息描述。F

13.Phong算法的计算量要比Gourand算法小得多。T

14.图形消隐中的深度存储算法，占用较多的内存空间，且未利用连贯性。F

# 五、简答题

**1.列举有关计算机图形学的应用电子出版及办公自动化**

计算机动画及广告、计算机艺术、科学计算的可视化、工业模拟、计算机辅助教学

**2.简要指出下列术语之间的主要不同之处：**

图形和图象：前者是指矢量表示的图，后者是指用点阵表示的图。

随机扫描与光栅扫描：前者是电子束扫描路径随图形不同而不同，后者是电子束扫描路径固定不变。

窗口与视口：前者是在世界（用户）坐标系中定义的，后者是设备坐标系中定义的。

数组与矩阵：前者是计算机语言中的术语，后者是数学中的术语。

逼近与插值：前者是所求曲线大致通过数据点，后者是所求曲线通过每一个数据点。

1. **图形交互技术有哪些？**

定位技术、定向技术、定路经技术、

定量技术、文本技术、橡皮筋技术、拖动技术

回显，引力域、草拟技术、旋转、形变、网格、约束

**4.列举有关计算机图形学的应用**

（1）计算机辅助设计与制造

（2）计算机辅助绘图

（3）计算机辅助教学

（4）办公自动化和电子出版技术

（5）计算机艺术

（6）在工业控制及交通方面的应用

（7）在医疗卫生方面的应用

（8）图形用户界面

**5.简述图形系统的组成与功能**

组成：计算机硬件、图形输入/输出设备、计算机系统软件和图形软件

功能：（1）计算功能（2）存储功能（3）交互功能（4）输入功能（5）输出功能

1. **DDA算法生成直线的基本原理：**

是从直线的微分方程中生成直线。由于直线的一阶导数是连续的，而且△ X和△ Y是成比例的，因此可以通过当前位置( Xi, Yi) 分别加上两个小增量

ε\*△ X和ε\*△ y (ε为无穷小的正数)来求出下一个点( Xi+1,Yi+1)的x，y的坐标，

即有： Xi+1= Xi +ε\*△ X；Yi+1= Yi +ε\*△ y

**7.计算机图形学研究的内容**

如何在计算机中表示图形以及利用计算机进行图形的计算，处理和显示的相关原理和算法

计算机动画 地理信息系统 人机交互 真实感图形显示 虚拟现实、科学计算可视化 并行图形处理

**8.简述点与多边形之间的包含性检测算法。**

先将该点变换到多边形所在平面，在二维空间讨论该问题，过该点沿任意方向做一条射线，与多边形边界求交点，若交点个数为偶数，0，2，4，?，则该点在多边形外部，否则在内部。如果射线与多边形某条边、某个顶点重合，则可以改变射线方向重新测试。

**9.简述边界表示法（BREP）实体构造表示法（CSG）。**

边界表示法是用实体的表面来表示实体的形状，它的基本元素是面、边、顶点。它以欧拉公式作为理论基础，要求实体的基本拓扑结构符合欧拉公式。目前采用的比较多的数据结构是翼边数据结构和半边数据结构（又称对称数据结构）。

实体构造表示法的基本思想是将简单的实体（又称体素）通过集合运算组成所需要的物体。其中，集合运算的实现过程由一棵二叉树来描述，二叉树的叶子节点表示体素或者几何变换的参数，非终端节点表示施加于其子结点的正则集合算子或几何变换的定义。

**10.为什么需要隐藏面消隐算法？Z缓冲区算法怎样判断哪个面应消隐？**

真实图形显示需要隐藏面消隐算法来判断哪些物体和表面遮挡了放在它们后面的物体和表面，从而产生更逼真的图像。Z缓冲区算法设置了一个二维数组，类似于帧缓冲器，里面存放的是每个像素点的深度值。Z缓冲器的初始值设置为某个大的数值，通常是后裁剪平面的距离。在判断像素（x,y）上的哪个平面更靠近观察者时，就可以简单地比较Z缓冲器中的深度值和当前平面的深度值。若当前平面的深度值小，则用新值替换Z缓冲器中的值，像素的颜色值也变成了新平面的颜色值。

Cohen-Sutherland裁剪过程

设直线p1，p2两端点代码分别为code1，code2。

则：

1. 若code1∪ code2=0，则直线在窗口内，全部显示（保留）

（2）若code1 ∩ code2≠0，说明两端点均在窗口外，应全部舍弃。

（3）若上述两种情况均不成立，则计算直线段与窗口边界的交点，在交点处把线段分为两段。其中一段完全在窗口外，可弃之。然后对另一段重复上述处理。

中点分割裁剪算法

* (1)输入直线段的两端点坐标。P1(x1,y1),p2(x2,y2),以及窗口的四条边 界坐标:wyt,wyb,wxl,wxr。
* (2)对p1,p2进行编码。P1,p2编码分别为code1,code2
* (3)若code1∪code2=0，则对直线段应该取之，保留当前直线段的端点坐标，转（5）；否则若code1&code2≠0，对直线段可弃之，转（5）。当上述两条件不满足时，进行步骤（4）。
* （4）求出直线段的中点M（x=round((x1+x2)/2） y=round((y1+y2)/2,）将P1M,P2M入栈。
* （5）当栈不空时，从栈中弹出一条直线段，取为p1,p2,转（2）进行处理，否则继续(6).
* (6)当栈为空时，合并保留的直线段端点，得到窗口内的直线段p1p2，用直线扫描法画出当前直线段p1p2,算法结束。

简述一个交互式计算机图形系统的功能有哪些

计算;存储;对话;输入;输出

一点透视步骤

1. 将三维形体平移到适当的位置l，m，n
2. 令视点在z轴，进行透视变换，此时透视变换矩阵

1000

0100

0011/d

0001

1. 最后向xOy面作正投影变换，将结果变换到xOy面上。

二点透视

1. 将三维形体平移到合适位置，使视点有一定高度切实形体的主要表面不会聚集成线
2. 将形体绕y轴旋转一个角度（<90°）方向满足右手定则
3. 进行透视变换，此时透视变换矩阵

100p

0100

001r

0001

4、最后向xOy面作正投影，即得二点透视图，这样可得道二点透视的变换矩阵。

# 六、程序题

请写出实现DDA生成直线算法（第一象限）的C（或者vc++）语言程序。

void DDALine(int x0,int y0,int x1,int y1,int color)

{

int dx,dy,epsl,k;

float x,y,xIncre,yIncre;

dx=x1-x0;

dy=y1-y0;

x=x0;y=y0;

if(abs(dx)>abs(dy))

epsl=abs(dx);

else

epsl=abs(dy);

xIncre=(float) dx/(float)epsl;

yIncre=(float) dy/(float)epsl;

for(k=0;k<=epsl;k++){

putpixel(int(x+0.5),(int)(y+0.5));

x+=xIncre;

y+=yIncre;

}

}

中点Bresenham算法

void MidBreshnhamLine(int x0,int y0,int x1,int y1,int color){

int dx,dy,d,UpIncre,DownIncre,x,y;

if(x0>x1){

x=x1;x1=x0;x0=x;

y=y1;y1=y0;y0=y;

}

x=x0;y=y0;

dx=x1-x0;dy=y1-y0;

d=dx-2\*dy;

UpIncre=2\*dx-2\*dy;DownIncre=-2\*dy;

while(x<=x1){

putpixel(x,y,color);

x++;

if(d<0){

y++;

d+=UpIncre;

}

else

d+=DownIncre;

}

}

Brsenham直线算法

void BreshnhamLine(int x0,int y0,int x1,int y1,int color){

int x,y，dx，dy,e;

dx=x1-x0;

dy=y1-y0;

e=-dx;x=x0;y=y0;

while(x<=x1){

putpixel(x,y,color);

x++;

e=e+2\*dy;

if(e>0){

y++;

e=e-2\*dx;

}

}

}

中点Breshnham画圆算法

void MidBreshnhamCircle(int r,int color)

{

int x,y,d;

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

while(x<=y){

CirclePoint(x,y,color);

if(d<0)

d+=2\*x+3;

else{

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

y--;

}

x++;

}

}

八分法画圆

void DrawEightPoint(CDC\* cc, int x, int y, int x0, int y0, COLORREF color)

{

//在绘制八分点的时候需要根据圆心进行坐标原点的平移

cc->SetPixel(x + x0, y + y0, color);//第一象限右下点

cc->SetPixel(x + x0, -y + y0, color);//第四象限右上点：与(x,y)关于x轴对称

cc->SetPixel(-x + x0, y + y0, color);//第二象限左下点：与(x,y)关于y轴对称

cc->SetPixel(-x + x0, -y + y0, color);//第三象限左上点：与(x,-y)关于y轴对称

cc->SetPixel(y + x0, x + y0, color);//第一象限左上点：与(x,y)关于y=x直线对称

cc->SetPixel(-y + x0, x + y0, color);//第二象限右上点：与(y,x)关于y轴对称

cc->SetPixel(-y + x0, -x + y0, color);//第三象限右下点：与(y,-x)关于x轴对称

cc->SetPixel(y + x0, -x + y0, color);//第四象限左下点：与(y,x)关于x轴对称

}

//中点Bresenham画圆算法（只能绘制八分之一圆弧）

void DrawCircle1(CDC\* cc, int x0, int y0, int radius, COLORREF color)

{

int x, y, d;

x = x0; y = y0 + radius; d = 1 - radius;

while (x <= y) {

//cc->SetPixel(x, y, RGB(0, 0, 0));

//根据第一个八分之弧上的点绘制其余的对应的七个点，并传入圆心坐标用于坐标原点和八分点的平移

DrawEightPoint(cc, x - x0, y - y0, x0, y0, color);

if (d < 0)d += 2 \* (x - x0) + 3;

else {

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

y--;

}

x++;

}

}

中点Breshnham画椭圆算法

void MidBreshnhamEllipse(int r,int color)

{

int x.y;

float d1,d2;

x=0;y=b;

d1=b\*b+a\*a\*(-b+0.25);

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

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

while(b\*b\*(x+1)<a\*a\*(y-0.5)){

if(d1<=0){

d1+=b\*b\*(2\*x+3);

x++;

}

else{

d1+=b\*b\*(2\*x+3)+a\*a\*(-2\*y+2);

x++;y--;

}

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

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

}

d2=b\*b\*(x+0.5)\*(x+0.5)+a\*a\*(y-1)\*(y-1)-a\*a\*b\*b;

while(y>0){

if(d2<=0){

d2+=b\*b\*(2\*x+2)+a\*a\*(-2\*y+3);

x++;y--;

}

else{

d2+=a\*a\*(-2\*y+3);

y--;

}

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

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

}

}

}

//平移变换

void CysxMFCView::OnPingyi()

{

// TODO: 在此添加命令处理程序代码

CDC\* cc = GetDC();

RedrawWindow();

DrawRectangle(cc, tx1, tx2, ty1, ty2);

DBianHuan bh;

bh.DoModal();

CPen pen;

pen.CreatePen(0, 1, RGB(255, 0, 0));

CPen\* oldpen = cc->SelectObject(&pen);

//操作方法

DrawRectangle(cc, tx1 + bh.PianX, tx2 + bh.PianX, ty1 + bh.PianY, ty2 + bh.PianY);

cc->SelectObject(oldpen);

}

//比例变换

void CysxMFCView::OnBili()

{

// TODO: 在此添加命令处理程序代码

CDC\* cc = GetDC();

RedrawWindow();

DrawRectangle(cc, 100, 200, 100, 200);

DBianHuan bh;

bh.DoModal();

CPen pen;

pen.CreatePen(0, 1, RGB(255, 0, 0));

CPen\* oldpen = cc->SelectObject(&pen);

//操作方法

int lengthX = (int)((float)(tx2 - tx1)\*bh.BiLi);

int lengthY = (int)((float)(ty2 - ty1)\*bh.BiLi);

DrawRectangle(cc, tx1, tx1 + lengthX, ty1, ty1 + lengthY);

cc->SelectObject(oldpen);

}

//顺时针旋转

int getBianHuanX(int x, int y, float angle)

{

return (int)((float)x\*cos(angle) + (float)y\*sin(angle));

}

int getBianHuanY(int x, int y, float angle)

{

return (int)(-(float)x\*sin(angle) + (float)y\*cos(angle));

}

//逆时针旋转

int getBianHuanX1(int x, int y, float angle)

{

return (int)((float)x\*cos(angle) - (float)y\*sin(angle));

}

int getBianHuanY1(int x, int y, float angle)

{

return (int)((float)x\*sin(angle) + (float)y\*cos(angle));

}

//绘制一个变换的四边形

void DrawBHRectangle(CDC\* cc, int x1, int x2, int y1, int y2, float angle, int x0, int y0)

{

//cc->MoveTo(x0+getBianHuanX(x1, y1,angle), y0+getBianHuanY(x1, y1,angle));

//cc->LineTo(x0 + getBianHuanX(x2, y1, angle), y0 + getBianHuanY(x2, y1, angle));

DrawLine1(cc, getBianHuanX(x1, y1, angle), getBianHuanY(x1, y1, angle), getBianHuanX(x2, y1, angle), getBianHuanY(x2, y1, angle), RGB(255, 0, 0));

//cc->MoveTo(x0 + getBianHuanX(x2, y1, angle), y0 + getBianHuanY(x2, y1, angle));

//cc->LineTo(x0 + getBianHuanX(x2, y2, angle), y0 + getBianHuanY(x2, y2, angle));

DrawLine1(cc, getBianHuanX(x2, y1, angle), getBianHuanY(x2, y1, angle), getBianHuanX(x2, y2, angle), getBianHuanY(x2, y2, angle), RGB(255, 0, 0));

//cc->MoveTo(x0 + getBianHuanX(x2, y2, angle), y0 + getBianHuanY(x2, y2, angle));

//cc->LineTo(x0 + getBianHuanX(x1, y2, angle), y0 + getBianHuanY(x1, y2, angle));

DrawLine1(cc, getBianHuanX(x2, y2, angle), getBianHuanY(x2, y2, angle), getBianHuanX(x1, y2, angle), getBianHuanY(x1, y2, angle), RGB(255, 0, 0));

//cc->MoveTo(x0 + getBianHuanX(x1, y2, angle), y0 + getBianHuanY(x1, y2, angle));

//cc->LineTo(x0 + getBianHuanX(x1, y1, angle), y0 + getBianHuanY(x1, y1, angle));

DrawLine1(cc, getBianHuanX(x1, y2, angle), getBianHuanY(x1, y2, angle), getBianHuanX(x1, y1, angle), getBianHuanY(x1, y1, angle), RGB(255, 0, 0));

}

//旋转变换

void CysxMFCView::OnXuanzhuan()

{

// TODO: 在此添加命令处理程序代码

CDC\* cc = GetDC();

RedrawWindow();

DrawRectangle(cc, tx1, tx2, ty1, ty2);

DBianHuan bh;

bh.DoModal();

//操作方法

//DrawBHRectangle(cc, tx1 - tx1, tx2 - tx1, ty1 - ty1, ty2 - ty1, bh.JiaoDu, tx1, ty1);

DrawBHRectangle(cc, tx1, tx2, ty1, ty2, HuDu(bh.JiaoDu), tx1, ty1);

}

void Draw4(CDC\* cc, POINT p1, POINT p2, POINT p3, POINT p4)

{

cc->MoveTo(p1);

cc->LineTo(p2);

cc->MoveTo(p2);

cc->LineTo(p3);

cc->MoveTo(p3);

cc->LineTo(p4);

cc->MoveTo(p4);

cc->LineTo(p1);

}

//对称变换

void CysxMFCView::OnDuichen()

{

// TODO: 在此添加命令处理程序代码

CDC\* cc = GetDC();

RedrawWindow();

DrawRectangle(cc, tx1, tx2, ty1, ty2);

DBianHuan bh;

bh.DoModal();

cc->MoveTo(bh.LineX1, bh.LineY1);

cc->LineTo(bh.LineX2, bh.LineY2);

cc->MoveTo(0, 0);

CPen pen;

pen.CreatePen(0, 1, RGB(255, 0, 0));

CPen\* oldpen = cc->SelectObject(&pen);

cc->SetViewportOrg(bh.LineX1, bh.LineY1);

//cc->LineTo(100, 400);//测试坐标原点是否移动

//操作方法

//定义四个点

POINT p1; p1.x = tx1; p1.y = ty1;

POINT p2; p2.x = tx2; p2.y = ty1;

POINT p3; p3.x = tx2; p3.y = ty2;

POINT p4; p4.x = tx1; p4.y = ty2;

//1.先平移(即将新点转移至新坐标系)

p1.x -= bh.LineX1; p1.y -= bh.LineY1;

p2.x -= bh.LineX1; p2.y -= bh.LineY1;

p3.x -= bh.LineX1; p3.y -= bh.LineY1;

p4.x -= bh.LineX1; p4.y -= bh.LineY1;

//2.再顺时针旋转

float angle = atan((float)(bh.LineY2 - bh.LineY1) / (float)(bh.LineX2 - bh.LineX1));

{

int x01 = getBianHuanX(p1.x, p1.y, angle), y01 = getBianHuanY(p1.x, p1.y, angle);

p1.x = x01; p1.y = y01;

int x02 = getBianHuanX(p2.x, p2.y, angle), y02 = getBianHuanY(p2.x, p2.y, angle);

p2.x = x02; p2.y = y02;

int x03 = getBianHuanX(p3.x, p3.y, angle), y03 = getBianHuanY(p3.x, p3.y, angle);

p3.x = x03; p3.y = y03;

int x04 = getBianHuanX(p4.x, p4.y, angle), y04 = getBianHuanY(p4.x, p4.y, angle);

p4.x = x04; p4.y = y04;

}

//3.再关于x轴对称

p1.y = -p1.y;

p2.y = -p2.y;

p3.y = -p3.y;

p4.y = -p4.y;

//4.旋转回去

{

int x01 = getBianHuanX1(p1.x, p1.y, angle), y01 = getBianHuanY1(p1.x, p1.y, angle);

p1.x = x01; p1.y = y01;

int x02 = getBianHuanX1(p2.x, p2.y, angle), y02 = getBianHuanY1(p2.x, p2.y, angle);

p2.x = x02; p2.y = y02;

int x03 = getBianHuanX1(p3.x, p3.y, angle), y03 = getBianHuanY1(p3.x, p3.y, angle);

p3.x = x03; p3.y = y03;

int x04 = getBianHuanX1(p4.x, p4.y, angle), y04 = getBianHuanY1(p4.x, p4.y, angle);

p4.x = x04; p4.y = y04;

}

//5.最后平移回去（转回原坐标系）

//p1.x += bh.LineX1; p1.y += bh.LineY1;

//p2.x += bh.LineX1; p2.y += bh.LineY1;

//p3.x += bh.LineX1; p3.y += bh.LineY1;

//p4.x += bh.LineX1; p4.y += bh.LineY1;

//绘制多边形

Draw4(cc, p1, p2, p3, p4);

cc->SetViewportOrg(-bh.LineX1, -bh.LineY1);

cc->SelectObject(oldpen);

}

//b代表x系数，即沿y方向错切；c代表y系数，即沿x方向错切

int CuoQieX(int x, int y, float b, float c)

{

if (b == 0)

{

return x + c\*y;

}

else if (c == 0)

{

return x;

}

else if (c != 0 && b != 0)

{

return x + c\*y;

}

return x;

}

int CuoQieY(int x, int y, float b, float c)

{

if (b == 0)

{

return y;

}

else if (c == 0)

{

return b\*x + y;

}

else if (c != 0 && b != 0)

{

return b\*x + y;

}

return y;

}

void CysxMFCView::OnCuoqie()

{

// TODO: 在此添加命令处理程序代码

CDC\* cc = GetDC();

RedrawWindow();

DrawRectangle(cc, 100, 200, 100, 200);

DBianHuan bh;

bh.DoModal();

CPen pen;

pen.CreatePen(0, 1, RGB(255, 0, 0));

CPen\* oldpen = cc->SelectObject(&pen);

//操作方法

float b = bh.CuoQieY, c = bh.CuoQieX;

POINT p1; p1.x = CuoQieX(tx1, ty1, b, c); p1.y = CuoQieY(tx1, ty1, b, c);

POINT p2; p2.x = CuoQieX(tx2, ty1, b, c); p2.y = CuoQieY(tx2, ty1, b, c);

POINT p3; p3.x = CuoQieX(tx2, ty2, b, c); p3.y = CuoQieY(tx2, ty2, b, c);

POINT p4; p4.x = CuoQieX(tx1, ty2, b, c); p4.y = CuoQieY(tx1, ty2, b, c);

Draw4(cc, p1, p2, p3, p4);

cc->SelectObject(oldpen);

}

**1.种子填充算法——递归算法可实现如下：**

void FloodFill4(int x,int y,int oldColor,int newColor)

{ if(GetPixel(x,y) == oldColor)

{ PutPixel(x,y,newColor);

FloodFill4(x,y+1,oldColor,newColor);

FloodFill4(x,y-1,oldColor,newColor);

FloodFill4(x-1,y,oldColor,newColor);

FloodFill4(x+1,y,oldColor,newColor);

}

}

**2.多边形的4种子填充算法——边界表示的4连通区域**

void CLine\_ScanView::BoundaryFill4(int x, int y, int boundarycolor, int newcolor)

{

CDC\* pDC = GetDC();

CPen pen;

pen.CreatePen(PS\_SOLID,2,RGB(0,255,0));

pDC->SelectObject(&pen);

int a[4][2] = {200,100,180,120,220,120,200,100};

pDC->MoveTo(a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

pDC->LineTo(a[i][0],a[i][1]);

}

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

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

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

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

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

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

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

}

}

**3.多边形的8种子填充算法——边界表示的4连通区域**

void CLine\_ScanView::BoundaryFill8(int x, int y, int boundarycolor, int newcolor)

{

CDC\* pDC = GetDC();

CPen pen;

pen.CreatePen(PS\_SOLID,2,RGB(0,255,0));

pDC->SelectObject(&pen);

int a[4][2] = {200,100,180,120,220,120,200,100};

pDC->MoveTo(a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

pDC->LineTo(a[i][0],a[i][1]);

}

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

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

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

BoundaryFill8(x,y+1,boundarycolor,newcolor);

BoundaryFill8(x+1,y+1,boundarycolor,newcolor);

BoundaryFill8(x,y-1,boundarycolor,newcolor);

BoundaryFill8(x-1,y-1,boundarycolor,newcolor);

BoundaryFill8(x+1,y,boundarycolor,newcolor);

BoundaryFill8(x+1,y-1,boundarycolor,newcolor);

BoundaryFill8(x-1,y,boundarycolor,newcolor);

BoundaryFill8(x-1,y+1,boundarycolor,newcolor);

}

}

**1.三维变换的比例变换响应函数**

void CLine\_ScanView::OnThreeProportion()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,255,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {2,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**2.三维变换的旋转变换响应函数**

void CLine\_ScanView::OnThreeRotating()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(255,0,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

int t = 600;

dc.SelectObject(&pen2);

dc.MoveTo (a[0][0]\*cos((t/6)\*1.0)-a[0][1]\*sin((t/6)\*1.0),//一个点

a[0][0]\*sin((t/6)\*1.0)+a[0][1]\*cos((t/6)\*1.0));

for(int i=0;i<4;i++) {

dc.LineTo (a[i][0]\*cos((t/6)\*1.0)-a[i][1]\*sin((t/6)\*1.0),//一个点

a[i][0]\*sin((t/6)\*1.0)+a[i][1]\*cos((t/6)\*1.0));

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**3.三维变换的错切变换响应函数**

void CLine\_ScanView::OnThreeShear()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(255,255,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {1,2,0,0,1.5,1,0,0,0,0,1,0,0,0,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**4.三维变换的对称变换响应函数**

void CLine\_ScanView::OnThreeSymmetry()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,255,255));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1}; //沿y=x对称

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**5.三维变换的平移变换响应函数**

void CLine\_ScanView::OnThreeTranslation()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,0,255));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {1,0,0,0,0,1,0,0,0,0,1,0,100,100,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

二维变换矩阵

平移

1 0 0

0 1 0

Tx Ty 1

比例

Sx 0 0

0 Sy 0

0 0 1

旋转

1 θ 0

-θ 1 0

0 0 0

对称---关于X

1 0 0

0 -1 0

0 0 1

对称---关于y

-1 0 0

0 1 0

0 0 1

对称---关于原点

-1 0 0

0 -1 0

0 0 1

对称---关于y=x

0 1 0

1 0 0

0 0 1

对称---关于y=-x

0 -1 0

-1 0 0

0 0 1

错切变换

1 b 0

c 1 0

0 0 1

三维变换

旋转变换—绕z

Cosθ sinθ 0 0

-sinθ cosθ 0 1

0 0 1 0

0 0 0 1

旋转变换—绕x

1 0 0 0

0 Cosθ sinθ 0

0 -sinθ cosθ 0

0 0 0 1

旋转变换—绕y

cosθ 0 -sinθ 0

0 1 0 0

-sinθ 0 cosθ 0

1. 0 0 1

对称于XOY平面



对称于YOZ平面



对称于XOZ平面



关于x轴对称



关于y轴对称



关于z轴对称



三维投影变换—正投影

主视图

1000

0000

0010

0001

俯视图

1000

0100

0000

0001

侧视图

0000

0100

0010

0001

斜投影

1000

0100

CotAcosB cotAsinB00

0001

透视投影

1000

0100

0011/d

0001