计算机图形学习题参考答案

第 1 章　绪论

1、第一届 ACM SIGGRAPH 会议是哪一年在哪里召开的？
解:
1974 年，在 Colorado 大学召开了第一届 SIGGRAPH 年会。

2、计算机图形学之父是谁？
解:
Sutherland

3、列举一些计算机图形学的应用领域（至少 5 个）。
解:
计算机辅助设计、图示图形学、计算机艺术、娱乐、教学与培训、可视化、图像处理、图形用户界面等。

4、简要介绍计算机图形学的研究内容。
解:
（1）图形的输入。如何开发和利用图形输入设备及相关软件把图形输入到计算机中，以便进行各种处理。
（2）图形的处理。包括对图形进行变换（如几何变换、投影变换）和运算（如图形的并、交、差运算）等处理。
（3）图形的生成和输出。如何将图形的特定表示形式转换成图形输出系统便于接受的表示形式，并将图形在显示器或打印机等输出设备上输出。

5、简要说明计算机图形学与相关学科的关系。
解:
与计算机图形学密切相关的学科主要有图像处理、计算几何、计算机视觉和模式识别等。计算机图形学着重讨论怎样将数据模型变成数字图像。图像处理着重研究图像的压缩存储和去除噪音等问题。模式识别重点讨论如何从图像中提取数据和模型。计算几何着重研究数据模型的建立、存储和管理。随着技术的发展和应用的深入，这些学科的界限变得模糊起来，各学科相互渗透、融合。一个较完善的应用系统通常综合利用了各个学科的技术。

6、简要介绍几种计算机图形学的相关开发技术。
解:
（1）OpenGL。OpenGL 是一套三维图形处理库，也是该领域事实上的工业标准。OpenGL 独立于硬件、操作系统和窗口系统，能运行于不同操作系统的各种计算机，并能在网络环境下以客户/服务器模式工作，是专业图形处理、科学计算等高端应用领域的标准图形库。以 OpenGL 为基础开发的应用程序可以十分方便地在各种平台间移植；OpenGL 与 C/C++紧密接合，便于实现图形的相关算法，并可保证算法的正确性和可靠性；OpenGL 使用简便，效率高。
（2）DirectX。DirectX 是一种图形应用程序接口，它是一个提高系统性能的加速软件，由微软公司创建开发。从内部原理来看，DirectX 是一系列 DLL，通过这些 DLL，程序员可以在无视设备差异的情况下访问底层硬件。DirectX 提供了一整套的多媒体接口方案，主要应用于游戏软件的开发。DirectX 给众多的软、硬件商家提供了一个共同开发的标准平台，硬件制造商按照该标准研发制造更好的产品，程序员根据这套标准开发游戏。
（3）Java 3D。Java 3D API 是 Sun 定义的用于实现三维显示的接口。Java 3D 把 OpenGL 和 DirectX 这些底层技术包装在 Java 接口中。这种全新的设计使三维技术变得不再繁琐并且可以加入到 J2SE、J2EE 的整套架构，这些特性保证了 Java 3D 技术强大的扩展性。Java 3D 主要实现了以下三维显示的功能：生成形体；使形体具有颜色、透明效果、贴图等；在三维环境中生成灯光、移动灯光等；具有行为处理判断能力（键盘、鼠标、定时等）；生成雾、背景、声音等；使形体变形、移动、生成三维动画等。
（4）VRML。VRML 是一种标记语言，而不是一种 API 开发包。它使用 VRML 浏览器能读懂的 ASCII 文本格式来描述世界和链接。VRML 既可以用来建立真实世界场景的模型，也可以用来建立虚构的三维世界。VRML 浏览器实际上是一个实时三维着色引擎，它使得 VRML 应用从三维建模和动画应用中分离出来，在三维建模和动画应用中可以预先对前方场景进行着色。VRML 提供了 6+1 个自由

度，可以沿着三个方向移动，也可以沿着三个方位旋转，同时还可以建立与其他三维空间的超链接。因此 VRML 是超空间的。

7、图形的构成要素有哪些？
解:
① 刻画形状的点、线、面、体等几何要素;
② 反映物体表面属性和材质的灰度、颜色等非几何要素。

8、计算机图形学的最高奖以谁的名字命名，获得第一届和第二届该奖的分别是谁？
解:
计算机图形学的最高奖是以 Coons 的名字命名的,获得第一届和第二届 Coons 奖的是 Ivan Sutherland 和 Pierre Bézier。

第 2 章　基本图元的显示

1、假设 RGB 光栅系统的设计采用 8×10 英寸的屏幕，每个方向的分辨率为每英寸 100 个像素。如果每个像素 6 位，存放在帧缓冲器中，则帧缓冲器需要多大存储容量（字节数）？
解:
$8\times100\times10\times100\times6/8=600000$（字节）。

2、假设计算机字长为 32 位，传输速率为 1 MIP（每秒百万条指令）。300 DPI(每英寸点数)的激光打印机，页面大小为 8.5×11 英寸，要填满帧缓冲器需要多长时间。
解:
$8.5\times300\times11\times300\times1/32/10^6=0.2630$（秒）。

3、考虑分辨率为 640×480 和 1280×1024 的两个光栅系统。若显示控制器刷新屏幕的速率为每秒 60 帧，各个系统每秒钟应访问为多少像素？各个系统每个像素的访问时间是多少？
解:
（1）每秒钟访问像素数为 $640\times480\times60=18432000$。每个像素的访问时间为 $1/18432000=5.4253\times10^{-8}$ 秒。
（2）每秒钟访问像素数为 $1208\times1024\times60=74219520$。每个像素的访问时间为 $1/74219520=1.3474\times10^{-8}$ 秒。

4、假设视频监视器的显示区域为 12×9.6 英寸。如果分辨率是 1280×1024，纵横比为 1，屏幕每点的直径是多少？
解:
$12/1280=0.0094$，$9.6/1024=0.0094$，所以屏幕每点的直径是 0.0094 英寸。

5、假设某真彩色（24 位）RGB 光栅系统有 800×600 的帧缓冲器，可用多少种不同的彩色选择（强度级）？在任一时刻可显示多少不同的彩色？
解:
强度等级为 2^{24} 种。每一时刻最多显示 $\min(2^{24},\ 800\times600)=480000$ 种不同颜色。

6、分辨率为 1024×768 的高质量彩色系统（32 位）至少需要多少 MB 帧缓冲器？
解:
$1024\times768\times32/8/1024/1024=3$（MB）

7、使用 DDA 算法绘制端点为 $(20,10)$ 和 $(30,18)$ 的线段。
解:

$\Delta x=10,\Delta y=8,m=0.8$

$x_0=20,y_0=10$	$x_6=26,y_6=y_5+m=14.8\approx15$
$x_1=21,y_1=y_0+m=10.8\approx11$	$x_7=27,y_7=y_6+m=15.6\approx16$
$x_2=22,y_2=y_1+m=11.6\approx12$	$x_8=28,y_8=y_7+m=16.4\approx16$
$x_3=23,y_3=y_2+m=12.4\approx12$	$x_9=29,y_9=y_8+m=17.2\approx17$
$x_4=24,y_4=y_3+m=13.2\approx13$	$x_{10}=30,y_{10}=y_9+m=18$
$x_5=25,y_5=y_4+m=14$	

8、使用中点算法绘制端点为$(20,10)$和$(30,18)$的线段。

解:

$a=-8, b=10, 2a=-16, (2a+2b)=4$

k	(x_k, y_k)	p_k
0	(20,10)	$2a+b=-6$
1	(21,11)	$p_0+(2a+2b)=-2$
2	(22,12)	$p_1+(2a+2b)=2$
3	(23,12)	$p_2+2a=-14$
4	(24,13)	$p_3+(2a+2b)=-10$
5	(25,14)	$p_4+(2a+2b)=-6$
6	(26,15)	$p_5+(2a+2b)=-2$
7	(27,16)	$p_6+(2a+2b)=2$
8	(28,16)	$p_7+2a=-14$
9	(29,17)	$p_8+(2a+2b)=-10$
10	(30,18)	

9、使用中点圆算法，绘制圆心为$(0,0)$，半径$r=10$的圆在第一象限中的部分。

解:

k	(x_k, y_k)	(x_k', y_k')	p_k
0	(0,10)	(10,0)	$1-r=-9$
1	(1,10)	(10,1)	$p_0+2x_1+1=-6$
2	(2,10)	(10,2)	$p_1+2x_2+1=-1$
3	(3,10)	(10,3)	$p_2+2x_3+1=6$
4	(4,9)	(9,4)	$p_3+2x_4+1-2y_4=-3$
5	(5,9)	(9,5)	$p_4+2x_5+1=8$
6	(6,8)	(8,6)	$p_5+2x_6+1-2y_6=5$
7	(7,7)		

10、使用中点椭圆算法，绘制中心为$(0,0)$，长半径$a=8$，短半径$b=6$的椭圆在第一象限中的部分。

解:

区域一（上半部分）

k	(x_k, y_k)	$2b^2x_k$	$2a^2y_k$	p_k
0	(0,8)	0	1600	$b^2-a^2b+(1/4)a^2=-332$
1	(1,8)	72	768	$p_0+2b^2x_1+b^2=-224$
2	(2,8)	144	768	$p_1+2b^2x_2+b^2=-44$
3	(3,8)	216	768	$p_2+2b^2x_3+b^2=208$
4	(4,7)	288	640	$p_3+2b^2x_4+b^2-2a^2y_4=-108$
5	(5,7)	360	640	$p_4+2b^2x_5+b^2=288$
6	(6,6)	432	512	$p_5+2b^2x_6+b^2-2a^2y_6=244$
7	(7,6)	504	384	
8	(8,5)			

区域二（下半部分）

k	(x_k, y_k)	p_k
0	(7,3)	$b^2(x_0+1/2)^2+a^2(y_0-1)^2-a^2b^2\approx-23$
1	(8,2)	$p_0-2a^2y_1+a^2+2b^2x_1=361$
2	(8,1)	$p_1-2a^2y_2+a^2=297$
3	(8,0)	

11、已知：$A(0,0)$、$B(1,1)$、$C(2,0)$、$D(1,2)$，请判断多边形$ABCD$是否是凹多边形。

解:

多边形的边向量为$\overrightarrow{AB}=(1,1,0)$，$\overrightarrow{BC}=(1,-1,0)$，$\overrightarrow{CD}=(-1,2,0)$，$\overrightarrow{DA}=(-1,-2,0)$。因为$\overrightarrow{AB}\times\overrightarrow{BC}=(0,0,-2)$，该多边形是凹多边形。

12、使用整数增量方法计算边$(1,1)\sim(7,6)$与各扫描线交点的x坐标。

解:

$\Delta x=6$，$\Delta y=5$，$2\Delta y=10$

$q=\Delta x \operatorname{idiv} \Delta y=1$，$q+1=2$，$r=\Delta x-q\Delta y=1$，$2r=2$

$y=1$， $k=0$， $x=1$

$y=2$， $k=2$， $x=x+1=2$

$y=3$， $k=4$， $x=x+1=3$

$y=4$， $k=6$， $x=x+2=5$， $k=6-10=-4$

$y=5$， $k=-2$， $x=x+1=6$

$y=6$， $k=0$， $x=x+1=7$

第 3 章　OpenGL 的基本图元

1、请写出 OpenGL 中指定点的大小和线宽的函数，要求写出完整的函数原型。
解:
void glPointSize (GLfloat size);
void glLineWidth (GLfloat width);

2、请写出 OpenGL 中启用点、线、面反走样的函数调用。
解:
glEnable(GL_POINT_SMOOTH);
glEnable(GL_LINE_SMOOTH);
glEnable(GL_POLYGON_SMOOTH);

3、请使用 OpenGL 和 GLUT 编写一个简单的图形程序，用于显示一个填充的白色矩形。其中矩形规定为(-0.8, -0.8) ~ (0.8, 0.8)，程序窗口的大小为(200, 200)，标题为“白色矩形”。
解:
```
#include <GL/glut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glRectf(-0.8, -0.8, 0.8, 0.8);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("白色矩形");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

4、请使用 OpenGL 和 GLUT 编写一个简单的图形程序，用于显示一个填充的红色三角形。其中三角形的顶点分别是(-0.8, -0.8)、(0.8, -0.8)和(0, 0.8)，程序窗口大小为(200, 200)，标题为“红色三角形”。
解:
```
#include <GL/glut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1, 0, 0);
    glBegin(GL_TRIANGLES);
    glVertex2d(-0.8, -0.8), glVertex2d(0.8, -0.8), glVertex2d(0, 0.8);
    glEnd();
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("红色三角形!");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

5、请使用 OpenGL 和 GLUT 编写一个简单的图形程序，用于演示点的反走样效果。要求使用线段(-0.6, -0.6) ~ (0.6, 0.6)上均匀分布的 5 个点（含端点），点的大小为 10 像素，程序窗口的大小为(200, 200)，

标题为“点的反走样”。
解:
```
#include <GL/glut.h>
void Paint()
{   double x;
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POINTS);
    for(x = - 0.6; x <= 0.6; x += 0.3) glVertex2f(x, x);
    glEnd();
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("点的反走样");
    glutDisplayFunc(Paint);
    glPointSize(10);
    glEnable(GL_POINT_SMOOTH);
    glutMainLoop();
}
```
6、请使用 OpenGL 和 GLUT 编写一个简单的图形程序，用于演示线段的反走样效果。其中线段的端点为(-0.6, -0.6)和(0.6, 0.6)，线宽为 5 像素，程序窗口的大小为(200, 200)，标题为“线段的反走样”。
解:
```
#include <GL/glut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINES);
    glVertex2f(-0.6, -0.6), glVertex2f(0.6, 0.6);
    glEnd();
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("线段的反走样");
    glutDisplayFunc(Paint);
    glLineWidth(5);
    glEnable(GL_LINE_SMOOTH);
    glutMainLoop();
}
```

第 4 章　二维图形变换

1、通过对 $R(\theta_1)$ 和 $R(\theta_2)$ 矩阵表示的旋转变换合并得到 $R(\theta_1)\times R(\theta_2)=R(\theta_1+\theta_2)$，证明两个复合的旋转是相加的。
解:

$$R(\theta_1)\times R(\theta_2)=\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 0\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0\\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0\\ 0 & 0 & 1\end{pmatrix}=R(\theta_1+\theta_2)$$

2、证明对下列每个操作序列来讲矩阵相乘是可以交换的。
（1）两个连续的旋转。
（2）两个连续的平移。
（3）两个连续的缩放。
解:
（1）因为

$$R(\theta_1)\times R(\theta_2)=\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 0\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0\\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R(\theta_2)\times R(\theta_1)=\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 0\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos(\theta_2+\theta_1) & -\sin(\theta_2+\theta_1) & 0\\ \sin(\theta_2+\theta_1) & \cos(\theta_2+\theta_1) & 0\\ 0 & 0 & 1\end{pmatrix}$$

所以 $R(\theta_1)\times R(\theta_2)=R(\theta_2)\times R(\theta_1)$。

（2）方法同（1）

（3）方法同（1）

3、证明一致缩放和旋转形成可交换的操作对，但通常缩放和旋转不是可交换的操作。

解:

（1）一致缩放与旋转的可交换性

$$S(s,s)\times R(\theta)=\begin{pmatrix}s & 0 & 0\\ 0 & s & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}s\cos\theta & -s\sin\theta & 0\\ s\sin\theta & s\cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R(\theta)\times S(s,s)=\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}s & 0 & 0\\ 0 & s & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}s\cos\theta & -s\sin\theta & 0\\ s\sin\theta & s\cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

所以，一致缩放和旋转是可交换的操作对。

（2）一般缩放和旋转不是可交换的：举例说明

$$S(1,2)\times R(60^\circ)=\begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1/2 & -\sqrt{3}/2 & 0\\ \sqrt{3}/2 & 1/2 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1/2 & -\sqrt{3}/2 & 0\\ \sqrt{3} & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R(60^\circ)\times S(1,2)=\begin{pmatrix}1/2 & -\sqrt{3}/2 & 0\\ \sqrt{3}/2 & 1/2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1/2 & -\sqrt{3} & 0\\ \sqrt{3}/2 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

所以，一般缩放和旋转不是可交换的操作对。

4、已知旋转角为θ，基准点位置为(x_r, y_r)，请构造该旋转变换的变换矩阵。

解:

（1）使基准点与原点重合：$T_1=T(-x_r,-y_r)$

（2）绕原点旋转：$R=R(\theta)$

（3）使基准点回到原处：$T_2=T(x_r,y_r)$

完整变换

$$M=T_2RT_1=\begin{pmatrix}1 & 0 & x_r\\ 0 & 1 & y_r\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -x_r\\ 0 & 1 & -y_r\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta & -\sin\theta & x_r(1\text{-}\cos\theta)+y_r\sin\theta\\ \sin\theta & \cos\theta & \text{-}x_r\sin\theta+y_r(1\text{-}\cos\theta)\\ 0 & 0 & 1\end{pmatrix}$$

5、已知缩放系数为s_x和s_y，固定点位置为(x_f, y_f)，请构造该缩放变换的变换矩阵。

解:

（1）使固定点与原点重合：$T_1=T(-x_r,-y_r)$

（2）以原点为固定点缩放：$S=S(s_x,s_y)$

（3）使固定点回到原处：$T_2=T(x_r,y_r)$

完整变换

$$M=T_2ST_1=\begin{pmatrix}1 & 0 & x_f\\ 0 & 1 & y_f\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -x_f\\ 0 & 1 & -y_f\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}s_x & 0 & x_f(1-s_x)\\ 0 & s_y & y_f(1-s_y)\\ 0 & 0 & 1\end{pmatrix}$$

6、确定反射轴为直线 $y=3x/4$ 的反射变换矩阵的形式。

解:

（1）使反射轴与 x 轴重合。因为反射轴通过原点，所以只需将向量 $(4,3)$ 变换为 x 轴单位向量即可。

$$\vec{u}=(4,3)/|(4,3)|=(0.8,0.6),\vec{v}=(-0.6,0.8)$$

$$R=\begin{pmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

（2）相对于 x 轴反射:

$$F_x=\begin{pmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

（3）使反射轴回到原处:

$$R^{-1}=\begin{pmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}^{-1}=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

完整变换:

$$M=R^{-1}\times F_x\times R=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.28 & 0.96 & 0\\ 0.96 & -0.28 & 0\\ 0 & 0 & 1\end{pmatrix}$$

7、请写出相对于 $y=0$ 沿 x 方向错切的变换矩阵，已知错切参数为: sh_x。

解:

$$\begin{cases}x'=x+y\times \mathrm{sh}_x\\ y'=y\end{cases}\qquad\begin{pmatrix}1 & \mathrm{sh}_x & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

8、已知 $P_0(3,3)$ 和 $P_1(6,7)$，新坐标系统的原点位置定义在旧坐标系统的 P_0 处，新的 y 轴为 P_0P_1，请构造完整的从旧坐标系统到新坐标系统的坐标变换矩阵。

解:

（1）使新原点与旧原点重合: $T=T(-3,-3)$

（2）使新坐标轴与旧坐标轴重合: R

$$\vec{v}=\overrightarrow{P_0P_1}/|\overrightarrow{P_0P_1}|=(0.6,0.8),\vec{u}=(0.8,-0.6)$$

$$R=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

完整变换:

$$M=RT=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & -3\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.8 & -0.6 & -0.6\\ 0.6 & 0.8 & -4.2\\ 0 & 0 & 1\end{pmatrix}$$

9、已知 $P_0(3,3)$ 和 $P_1(6,7)$，新坐标系统的原点位置定义在旧坐标系统的 P_0 处，新的 x 轴为 P_0P_1，请构造完整的从旧坐标系统到新坐标系统的坐标变换矩阵。

解:

（1）使新原点与旧原点重合: $T=T(-3,-3)$

（2）使新坐标轴与旧坐标轴重合: R

$$\vec{u}=\overrightarrow{P_0P_1}/|\overrightarrow{P_0P_1}|=(0.6,0.8),\ \vec{v}=(-0.8,0.6)$$

$$R=\begin{pmatrix}0.6 & 0.8 & 0\\ -0.8 & 0.6 & 0\\ 0 & 0 & 1\end{pmatrix}$$

完整变换：

$$M=RT=\begin{pmatrix}0.6 & 0.8 & 0\\ -0.8 & 0.6 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & -3\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.6 & 0.8 & -4.2\\ -0.8 & 0.6 & 0.6\\ 0 & 0 & 1\end{pmatrix}$$

10、已知 $P_0(3,3)$ 和 $P_1(6,7)$，请构造一个变换，使 P_0P_1 与 x 轴重合。

解：

（1）使 P_0 与原点重合：$T=T(-3,-3)$

（2）使 P_0P_1 与 x 轴重合：R

$$\vec{u}=\overrightarrow{P_0P_1}\,/\,|\overrightarrow{P_0P_1}|=(0.6,0.8),\ \vec{v}=(-0.8,0.6)$$

$$R=\begin{pmatrix}0.6 & 0.8 & 0\\ -0.8 & 0.6 & 0\\ 0 & 0 & 1\end{pmatrix}$$

完整变换：

$$M=RT=\begin{pmatrix}0.6 & 0.8 & 0\\ -0.8 & 0.6 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & -3\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.6 & 0.8 & -4.2\\ -0.8 & 0.6 & 0.6\\ 0 & 0 & 1\end{pmatrix}$$

11、已知 $P_0(3,3)$ 和 $P_1(6,7)$，请构造一个变换，使 P_0P_1 与 y 轴重合。

解：

（1）使 P_0 与原点重合：$T=T(-3,-3)$

（2）使 P_0P_1 与 y 轴重合：R

$$\vec{v}=\overrightarrow{P_0P_1}\,/\,|\overrightarrow{P_0P_1}|=(0.6,0.8),\vec{u}=(0.8,-0.6)$$

$$R=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

完整变换：

$$M=RT=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & -3\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.8 & -0.6 & -0.6\\ 0.6 & 0.8 & -4.2\\ 0 & 0 & 1\end{pmatrix}$$

12、已知旋转角为 $60°$，基准点位置为 $(1,2)$，请构造该旋转变换的变换矩阵 M，结果至少保留 3 位小数。

解：

（1）使基准点与原点重合：$T_1=T(-1,-2)$

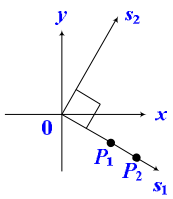
（2）绕原点旋转：$R=R(60°)$

（3）使基准点回到原处：$T_2=T(1,2)$

完整变换

$$M=T_2RT_1=\begin{pmatrix}1 & 0 & 1\\ 0 & 1 & 2\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos 60° & -\sin 60° & 0\\ \sin 60° & \cos 60° & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -1\\ 0 & 1 & -2\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}0.5 & -0.866 & 2.2321\\ 0.866 & 0.5 & 0.1340\\ 0 & 0 & 1\end{pmatrix}$$

13、在如图所示的方向上，用 s_1 和 s_2 作为缩放系数，请构造完成这种缩放的变换矩阵，其中固定点为原点，$s_1=2$，$s_2=3$，且方向 s_1 上两点的坐标分别为 $(2,-1)$ 和 $(6,-4)$。



第 13 题

解：

（1）旋转使 s_1 和 s_2 的方向分别与 x 和 y 轴重合：R

将s_1的方向$(6-2,-4-(-1))=(4,3)$单位化，得$\vec{u}=(0.8,-0.6)$；

令$\vec{v}=(0.6,0.8)$，从而

$$R=\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

（2）缩放：$S=S(2,3)$

（3）反向旋转使s_1和s_2的方向回到原始位置：R^{-1}

完整变换

$$M=R^{-1}SR=\begin{pmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}2 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}2.36 & 0.48 & 0\\ 0.48 & 2.64 & 0\\ 0 & 0 & 1\end{pmatrix}$$

14、已知窗口为$(0,0)\sim(10,10)$，视区为$(1,1)\sim(6,6)$，要将窗口中位于(x_w,y_w)的点映像到视区中坐标为(x_v,y_v)的点，请构造变换公式和变换矩阵。

解:

为了使视区与窗口中的对象有同样的相对位置，必须满足:

$$\begin{cases}\dfrac{x_v-1}{6-1}=\dfrac{x_w-0}{10-0}\\ \dfrac{y_v-1}{6-1}=\dfrac{y_w-0}{10-0}\end{cases}$$

从而，得到如下变换公式和变换矩阵

$$\begin{cases}x_v=0.5x_w+1\\ y_v=0.5y_w+1\end{cases},\quad\begin{pmatrix}0.5 & 0 & 1\\ 0 & 0.5 & 1\\ 0 & 0 & 1\end{pmatrix}$$

15、已知线段P_1P_2的两个端点坐标为$P_1(-5,10)$和$P_2(10,-5)$，裁剪窗口为$(0,0)\sim(10,10)$，请使用Cohen-Sutherland线段裁剪算法计算出裁剪以后剩余的线段。

解:

左边界：$x=0$，右边界：$x=10$，下边界：$y=0$，上边界：$y=10$。

P_1区域码：0001，P_2区域码：0100。两端点区域码的与为0000。

P_1是一外端点，位于窗口左边，P_1P_2与左边界$x=0$求交

$$m=(-5-10)/(10+5)=-1$$
$$y=y_1+m(c-x_1)=10-(0+5)=5$$

得交点$P_1'=(0,5)$。舍弃P_1P_1'，保留$P_1'P_2$。

P_1'区域码：0000，P_2区域码：0100。两端点区域码的与为0000。

P_2是一外端点，位于窗口的下边，$P_1'P_2$与下边界$y=0$求交

$$1/m=-1,\quad x=x_1+(c-y_1)/m=-5-(0-10)=5$$

得交点$P_2'=(5,0)$。舍弃$P_2'P_2$，保留$P_1'P_2'$。

P_1'区域码：0000，P_2'区域码：0000。所以裁剪后剩余线段为$P_1'P_2'$，端点坐标分别为：$P_1'=(0,5)$，$P_2'=(5,0)$。

16、已知三角形的顶点分别为：$A(-1,0)$、$B(1,2)$和$C(3,0)$，裁剪窗口为：$(0,1)\sim(3,3)$，请使用Sutherland-Hodgeman多边形裁剪算法计算出该多边形被裁剪以后剩余的部分。

解：初始顶点集合：$V_0=\{A,B,C\}$。

① 左裁剪。顶点集合$V_1=\varnothing$。

- 对AB，A在左边界外，B在左边界内，且AB与左边界的交点为$A'=(0,1)$，故$V_1=\{A',B\}$。

- 对BC，B和C都均在左边界内，故$V_1=\{A', B, C\}$。
- 对CA，C在左边界内，A在左边界外，CA与左边界的交点$C'=(0,0)$，故$V_1=\{A', B, C, C'\}$。

② 右裁剪。A'、B、C和C'均在右边界内，故顶点集合$V_2=\{A', B, C, C'\}$。

③ 下裁剪。顶点集合$V_3=\varnothing$。
- 对$A'B$，A'和B均在下边界内，故$V_3=\{B\}$。
- 对BC，B在下边界内，C在下边界外，且BC与下边界的交点为$B'=(2,1)$，故$V_3=\{B, B'\}$。
- 对CC'，C和C'均在下边界外，故$V_3=\{B, B'\}$。
- 对$C'A'$，C'在下边界外，A'在下边界内，且$C'A'$与下边界的交点$A'=(0,1)$，故$V_3=\{B, B', A', A'\}=\{B, B', A'\}$。

（4）上裁剪。B、B'和A'均在上边界内，故顶点集合$V_4=\{B, B', A'\}$。

所以，裁剪以后的多边形为$BB'A'$，其中$B=(1,2)$，$B'=(2,1)$，$A'=(0,1)$。

第 5 章　三维图形变换

1、已知三个顶点$V_1(1,2,1)$、$V_2(3,4,2)$和$V_3(2,5,3)$，从里向外以右手系形成逆时针方向。请构造出这三个顶点所确定的平面方程。

解:

$\overrightarrow{V_1V_2}=(2,2,1)$, $\overrightarrow{V_1V_3}=(1,3,2)$, $\vec{N}=\overrightarrow{V_1V_2}\times\overrightarrow{V_1V_3}=(1,-3,4)$，设平面方程为：$\vec{N}\bullet P+D=0$，即$x-3y+4z+D=0$。

将V_1代入该方程，有$1-3\times2+4+D=0$，从而$D=1$。所以该平面的方程为$x-3y+4z+1=0$。

2、已知旋转轴为z轴，旋转角为θ。请使用齐次坐标写出该旋转变换的变换矩阵和变换方程。

解:

$$\begin{cases} x'=x\cos\theta-y\sin\theta \\ y'=x\sin\theta+y\cos\theta \\ z'=z \end{cases} \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}=\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3、已知：$P_0(3,3,5)$和$P_1(6,7,5)$，旋转轴为P_0P_1，旋转角为θ。请使用齐次坐标写出该旋转变换的变换矩阵和变换方程。

解:

（1）使P_0与原点重合：$T_1=T(-3,-3,-5)$

（2）使P_0P_1与z轴重合：R_1

$\vec{N}=\overrightarrow{P_0P_1}=(3,4,0)$，$\vec{V}=\vec{N}\times(1,0,0)=(0,0,-4)$，$\vec{U}=\vec{V}\times\vec{N}=(16,-12,0)$。

将$\vec{U}$, $\vec{V}$, $\vec{N}$单位化，得$\vec{u}=\vec{U}/|\vec{U}|=(0.8,-0.6,0)$，$\vec{v}=\vec{V}/|\vec{V}|=(0,0,-1)$，$\vec{n}=\vec{N}/|\vec{N}|=(0.6,0.8,0)$。由此可得

$$R_1=\begin{pmatrix} 0.8 & -0.6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0.6 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

（3）绕z轴旋转：$R=R(\theta)$

（4）使P_0P_1方向复原：$R_2=R_1^{-1}$

（5）使P_0P_1回到原处：$T_2=T(3,3,5)$

完整变换

$$M=T_2\times R_2\times R\times R_1\times T_1$$
$$=\begin{pmatrix}1&0&0&3\\0&1&0&3\\0&0&1&5\\0&0&0&1\end{pmatrix}\begin{pmatrix}0.8&0&0.6&0\\-0.6&0&0.8&0\\0&-1&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}\cos\theta&-\sin\theta&0&0\\\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0.8&-0.6&0&0\\0&0&-1&0\\0.6&0.8&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-3\\0&1&0&-3\\0&0&1&-5\\0&0&0&1\end{pmatrix}$$
$$=\begin{pmatrix}0.64\cos\theta+0.36&0.48(1-\cos\theta)&0.8\sin\theta&0.48(1-\cos\theta)-4\sin\theta\\0.48(1-\cos\theta)&0.36\cos\theta+0.64&-0.6\sin\theta&-0.36(1-\cos\theta)+3\sin\theta\\-0.8\sin\theta&0.6\sin\theta&\cos\theta&5(1-\cos\theta)+0.6\sin\theta\\0&0&0&1\end{pmatrix}$$

4、已知缩放系数为 s_x、s_y 和 s_z，固定点位置为 (x_f, y_f, z_f)，请构造该缩放变换的变换矩阵。

解:

① 使固定点与原点重合：$T_1=T(-x_f,-y_f,-z_f)$。

② 相对于原点缩放：$S=S(s_x, s_y, s_z)$。

③ 使固定点回到原来位置：$T_2=T(x_f, y_f, z_f)$。

完整变换:

$$M=T_2\times S\times T_1=\begin{pmatrix}1&0&0&x_f\\0&1&0&y_f\\0&0&1&z_f\\0&0&0&1\end{pmatrix}\begin{pmatrix}s_x&0&0&0\\0&s_y&0&0\\0&0&s_z&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-x_f\\0&1&0&-y_f\\0&0&1&-z_f\\0&0&0&1\end{pmatrix}=\begin{pmatrix}s_x&0&0&(1-s_x)x_f\\0&s_y&0&(1-s_y)y_f\\0&0&s_z&(1-s_z)z_f\\0&0&0&1\end{pmatrix}$$

5、请写出相对于 $z=0$ 沿 x、y 方向错切的变换方程和变换矩阵。

解:

$$\begin{cases}x'=x+az\\y'=y+bz\\z'=z\end{cases}\Rightarrow SH_z=\begin{pmatrix}1&0&a&0\\0&1&b&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

6、已知新坐标系统的原点位置定义在旧坐标系的 (x_0, y_0, z_0) 处，且单位轴向量分别为 $\vec{u}$、$\vec{v}$ 和 $\vec{n}$，分别对应新的 x、y 和 z 轴，请构造完整的从旧坐标系到新坐标系的坐标变换矩阵。其中 $\vec{u}=(u_1, u_2, u_3)$、$\vec{v}=(v_1, v_2, v_3)$、$\vec{n}=(n_1, n_2, n_3)$。

解:

① 使新坐标系的原点与旧坐标系的原点重合：$T=T(-x_0,-y_0,-z_0)$

② 使新坐标系的坐标轴与旧坐标系的坐标轴重合：R

$$R=\begin{pmatrix}u_1&u_2&u_3&0\\v_1&v_2&v_3&0\\n_1&n_2&n_3&0\\0&0&0&1\end{pmatrix}$$

完整变换:

$$M=RT=\begin{pmatrix}u_1&u_2&u_3&0\\v_1&v_2&v_3&0\\n_1&n_2&n_3&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-x_0\\0&1&0&-y_0\\0&0&1&-z_0\\0&0&0&1\end{pmatrix}=\begin{pmatrix}u_1&u_2&u_3&-u_1x_0-u_2y_0-u_3z_0\\v_1&v_2&v_3&-v_1x_0-v_2y_0-v_3z_0\\n_1&n_2&n_3&-n_1x_0-n_2y_0-n_3z_0\\0&0&0&1\end{pmatrix}$$

7、已知在 $OXYZ$ 坐标系中某个平面的方程为 $3x+4y-10=0$，试求变换矩阵 M，使该平面在 $O_1X_1Y_1Z_1$ 坐标系下变成 $z_1=0$ 的平面。

解:

平面法向量在旧坐标系中的坐标为 $(3,4,0)$，且点 $(2,1,0)$ 在该平面上，而平面法向量在新坐标系中的

坐标为$(0,0,1)$。我们可将点$(2,1,0)$作为新原点，$(3,4,0)$作为新z轴方向，使用坐标系变换来完成该变换。

① 使新原点与旧原点重合：$T=T(-2,-1,0)$

② 使新坐标轴与旧坐标轴重合：R

- 新z方向：$\overrightarrow{N}=(3,4,0)$
- 新y方向：$\overrightarrow{V}=\overrightarrow{N}\times\vec{u}_x=\overrightarrow{N}\times(1,0,0)=(0,0,-4)$
- 新x方向：$\overrightarrow{U}=\overrightarrow{V}\times\overrightarrow{N}=(16,-12,0)$

将$\overrightarrow{U}$、$\overrightarrow{V}$、$\overrightarrow{N}$单位化，得$\vec{u}=\overrightarrow{U}/|\overrightarrow{U}|=(0.8,-0.6,0)$，$\vec{v}=\overrightarrow{V}/|\overrightarrow{V}|=(0,0,-1)$，$\vec{n}=\overrightarrow{N}/|\overrightarrow{N}|=(0.6,0.8,0)$，从而

$$R=\begin{pmatrix}0.8&-0.6&0&0\\0&0&-1&0\\0.6&0.8&0&0\\0&0&0&1\end{pmatrix}$$

完整变换：

$$M=RT=\begin{pmatrix}0.8&-0.6&0&0\\0&0&-1&0\\0.6&0.8&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-2\\0&1&0&-1\\0&0&1&0\\0&0&0&1\end{pmatrix}=\begin{pmatrix}0.8&-0.6&0&-1\\0&0&-1&0\\0.6&0.8&0&-2\\0&0&0&1\end{pmatrix}$$

8、已知旋转轴为AB，其中$A=(0,0,0)$，$B=(3,4,0)$，请构造绕AB旋转90度的旋转变换。

（1）旋转物体使旋转轴与某一坐标轴(通常取z轴)重合：R

将$\overrightarrow{AB}=(3,4,0)$单位化，得$\vec{n}=\overrightarrow{AB}/|\overrightarrow{AB}|=(0.6,0.8,0)$

令$\vec{u}_x=(1,0,0)$，$\vec{v}=\vec{n}\times\vec{u}_x/|\vec{n}\times\vec{u}_x|=(0.6,0.8,0)\times(1,0,0)/|(0.6,0.8,0)\times(1,0,0)|=(0,0,-1)$，$\vec{u}=\vec{v}\times\vec{n}=(0,0,-1)\times(0.6,0.8,0)=(0.8,-0.6,0)$，则

$$R=\begin{pmatrix}0.8&-0.6&0&0\\0&0&-1&0\\0.6&0.8&0&0\\0&0&0&1\end{pmatrix}$$

（2）绕坐标轴（z轴）完成指定的旋转：$R_z(\theta)$

（3）使旋转轴回到原来的方向：R^{-1}

完整变换：

$$M=R^{-1}\times R_z(90^\circ)\times R=\begin{pmatrix}0.8&0.6&0&0\\0&0&-1&0\\-0.6&0.8&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0&-1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0.8&-0.6&0&0\\0&0&-1&0\\0.6&0.8&0&0\\0&0&0&1\end{pmatrix}=\begin{pmatrix}0.36&0.48&0.8&0\\0.48&0.64&-1&0\\-0.8&0.6&0&0\\0&0&0&1\end{pmatrix}$$

9、已知旋转角为60°，旋转轴为$\begin{cases}x=1\\y=2\end{cases}$，请构造该三维旋转变换的变换矩阵$M$，结果至少保留3位小数。

解：

（1）使旋转轴与z重合，只需使位置$(1,2,0)$与原点重合即可：$T_1=T(-1,-2,0)$

（2）绕z旋转：$R=R(60^\circ)$

（3）使旋转轴回到原处：$T_2=T(1,2,0)$

完整变换

$$M=T_2RT_1=\begin{pmatrix}1&0&0&1\\0&1&0&2\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}\cos 60^\circ&-\sin 60^\circ&0&0\\\sin 60^\circ&\cos 60^\circ&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-1\\0&1&0&-2\\0&0&1&0\\0&0&0&1\end{pmatrix}=\begin{pmatrix}0.5&-0.866&0&2.2321\\0.866&0.5&0&0.1340\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

10、已知：观察参考点 $P(1,1,1)$，观察平面法向量 $\overrightarrow{N}=(4,3,0)$，观察向上向量 $\overrightarrow{V}=(-3,4,0)$。请构造从世界坐标到观察坐标的变换，写出变换矩阵。

解：

（1）使观察参考点与世界坐标系原点重合： $T=T(-1,-1,-1)$

（2）使观察坐标系与世界坐标系重合： R

观察坐标系 z 轴方向： $\overrightarrow{N}=(4,3,0)$

观察坐标系 x 轴方向： $\overrightarrow{U}=\overrightarrow{V}\times\overrightarrow{N}=(0,0,-25)$

观察坐标系 y 轴方向： $\overrightarrow{V}=\overrightarrow{N}\times\overrightarrow{U}=(-75,100,0)$

将 $\overrightarrow{U}$ 、 $\overrightarrow{V}$ 、 $\overrightarrow{N}$ 单位化：

$$\vec{u}=\overrightarrow{U}/|\overrightarrow{U}|=(0,0,-1),\quad \vec{v}=\overrightarrow{V}/|\overrightarrow{V}|=(-0.6,0.8,0),\quad \vec{n}=\overrightarrow{N}/|\overrightarrow{N}|=(0.8,0.6,0)$$

$$R=\begin{pmatrix}0&0&-1&0\\-0.6&0.8&0&0\\0.8&0.6&0&0\\0&0&0&1\end{pmatrix}$$

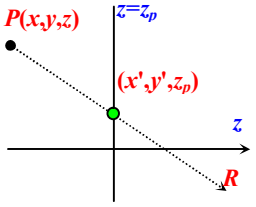
完整变换：

$$M=RT=\begin{pmatrix}0&0&-1&0\\-0.6&0.8&0&0\\0.8&0.6&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&-1\\0&1&0&-1\\0&0&1&-1\\0&0&0&1\end{pmatrix}=\begin{pmatrix}0&0&-1&1\\-0.6&0.8&0&-0.2\\0.8&0.6&0&-1.4\\0&0&0&1\end{pmatrix}$$

11、已知投影向量为 $\overrightarrow{V}=(3,4,5)$，投影面为 $z=0$，请根据定义计算该平行投影的变换矩阵。

解：

如图，投影线 PR 的参数方程为



$$\begin{cases}x'=x+3u\\y'=y+4u\\z'=z+5u\end{cases}\quad 0\le u<+\infty$$

将 $z'=0$ 代入上述方程，得 $u=(0-z)/5=-0.2z$。

于是得到该平行影变换的方程为

$$\begin{cases}x_p=x-0.6z\\y_p=y-0.8z\end{cases}$$

原始 z 坐标作为深度信息保存。

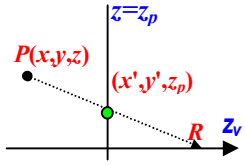
从而，变换矩阵为：

$$M=\begin{pmatrix}1&0&-0.6&0\\0&1&-0.8&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

12、求经过透视投影变换后点 $P(1,2,3)$ 的坐标。已知：观察平面为 $z=4$，投影中心为 $(0,0,5)$。

解：

如图，透视线 PR 的参数方程为



$$\begin{cases}x=1-u\\y=2-2u\\z=3+2u\end{cases}\quad 0\le u\le 1$$

将 $z=4$ 代入上述方程，得 $u=1/2$ 。

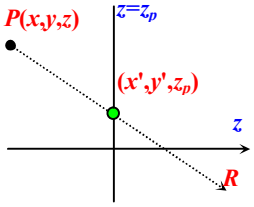
$$\begin{cases}x=1/2\\y=1\end{cases}$$

所以经过透视变换后，点 P 的坐标变为 $(1/2,1)$ 。

13、已知投影面为 $z=z_p$ ，投影中心为 $R(x_r,y_r,z_r)$ ，求透视投影变换矩阵。

解:

如图，透视线 PR 的参数方程为



$$\begin{cases}x'=x+(x_r-x)u\\y'=y+(y_r-y)u\\z'=z+(z_r-z)u\end{cases}\quad 0\le u\le 1$$

将 $z'=z_p$ 代入上述方程，得 $u=(z_p-z)/(z_r-z)$ 。

于是得到透视变换方程

$$\begin{cases}x_p=\dfrac{z_r-z_p}{z_r-z}x+\dfrac{z_p-z}{z_r-z}x_r\\y_p=\dfrac{z_r-z_p}{z_r-z}y+\dfrac{z_p-z}{z_r-z}y_r\end{cases}$$

令 $h=(z_r-z)$ ，则

$$\begin{cases}x_h=h\bullet x_p=(z_r-z)\left(\dfrac{z_r-z_p}{z_r-z}x+\dfrac{z_p-z}{z_r-z}x_r\right)=(z_r-z_p)x-x_rz+z_px_r\\y_h=h\bullet y_p=(z_r-z)\left(\dfrac{z_r-z_p}{z_r-z}y+\dfrac{z_p-z}{z_r-z}y_r\right)=(z_r-z_p)y-y_rz+z_py_r\\z_h=h\bullet z_p=(z_r-z)z_p=-z_pz+z_pz_r\\h=z_r-z=-z+z_r\end{cases}$$

由此可得如下矩阵形式的透视变换:

$$\begin{pmatrix}x_h\\y_h\\z_h\\h\end{pmatrix}=\begin{pmatrix}z_r-z_p & 0 & -x_r & z_px_r\\0 & z_r-z_p & -y_r & z_py_r\\0 & 0 & -z_p & z_pz_r\\0 & 0 & -1 & z_r\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$$

然后使用公式 $x_p=x_h/h,\ y_p=y_h/h$ 由齐次坐标计算投影坐标。

/*　或者:

令 $h=(z_r-z)/(z_r-z_p)$ ，则

$$\begin{cases}x_h=h\bullet x_p=\dfrac{z_r-z}{z_r-z_p}\left(\dfrac{z_r-z_p}{z_r-z}x+\dfrac{z_p-z}{z_r-z}x_r\right)=x+\dfrac{-x_r}{z_r-z_p}z+\dfrac{z_px_r}{z_r-z_p}\\y_h=h\bullet y_p=\dfrac{z_r-z}{z_r-z_p}\left(\dfrac{z_r-z_p}{z_r-z}y+\dfrac{z_p-z}{z_r-z}y_r\right)=y+\dfrac{-y_r}{z_r-z_p}z+\dfrac{z_py_r}{z_r-z_p}\\z_h=h\bullet z_p=\dfrac{z_r-z}{z_r-z_p}z_p=\dfrac{-z_p}{z_r-z_p}z+\dfrac{z_pz_r}{z_r-z_p}\\h=\dfrac{z_r-z}{z_r-z_p}=\dfrac{-1}{z_r-z_p}z+\dfrac{z_r}{z_r-z_p}\end{cases}$$

由此可得如下矩阵形式的透视变换:

$$\begin{pmatrix} x_h \\ y_h \\ z_h \\ h \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{-x_r}{z_r - z_p} & \frac{z_p x_r}{z_r - z_p} \\ 0 & 1 & \frac{-y_r}{z_r - z_p} & \frac{z_p y_r}{z_r - z_p} \\ 0 & 0 & \frac{-z_p}{z_r - z_p} & \frac{z_p z_r}{z_r - z_p} \\ 0 & 0 & \frac{-1}{z_r - z_p} & \frac{z_r}{z_r - z_p} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

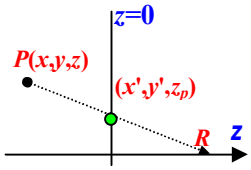
然后使用公式 $x_p = x_h / h,\ y_p = y_h / h$ 由齐次坐标计算投影坐标。

*/

14、已知投影面为 xy，投影中心为 $(0, 0, r)$，求透视投影变换矩阵。

解:

如图，透视线 PR 的参数方程为



$$\begin{cases} x' = x - xu \\ y' = y - yu \\ z' = z + (r - z)u \end{cases} \quad 0 \le u \le 1$$

将 $z' = 0$ 代入上述方程，得 $u = -z/(r - z)$。

$$\begin{cases} x' = x\left(\frac{r}{r - z}\right) \\ y' = y\left(\frac{r}{r - z}\right) \end{cases}$$

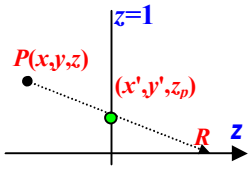
令 $h = (r - z)/r$，则 $x_h = x' \times h = x$，$y_h = y' \times h = y$，$z_h = 0$，$h = -z/r + 1$。所以透视变换矩阵为:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/r & 1 \end{pmatrix}$$

15、已知投影中心为原点，投影面为 $z = 1$，请根据定义计算该透视投影的变换矩阵。

解:

如图，透视线 PR 的参数方程为



$$\begin{cases} x' = x + (0 - x)u = x(1 - u) \\ y' = y + (0 - y)u = y(1 - u) \\ z' = z + (0 - z)u = z(1 - u) \end{cases} \quad 0 \le u \le 1$$

将 $z' = 1$ 代入上述方程，得 $1 - u = 1/z$。

$$\begin{cases} x' = x/z \\ y' = y/z \end{cases}$$

令 $h = z$，则 $x_h = x' \times h = x$，$y_h = y' \times h = y$，$z_h = z' \times h = z$。所以，变换矩阵为

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

16、已知矩形管道观察体为 $(1,1,1) \sim (10,10,10)$，规范化立方体为 $(-1,-1,-1) \sim (1,1,1)$，要将窗口中位于 (x, y, z) 的点映射到规范化立方体坐标为 (x', y', z') 的点，请构造变换公式和变换矩阵。

解:

为了使观察体和立方体中的对象有同样的相对位置，必须满足:

$$\frac{x' - (-1)}{1 - (-1)} = \frac{x - 1}{10 - 1},\ \frac{y' - (-1)}{1 - (-1)} = \frac{y - 1}{10 - 1},\ \frac{z' - (-1)}{1 - (-1)} = \frac{z - 1}{10 - 1}$$

从而，得到如下变换公式和变换矩阵

$$\begin{cases} x'=2x/9-11/9 \\ y'=2y/9-11/9 \\ z'=2z/9-11/9 \end{cases} \Rightarrow \begin{pmatrix} 2/9 & 0 & 0 & -11/9 \\ 0 & 2/9 & 0 & -11/9 \\ 0 & 0 & 2/9 & -11/9 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

第 6 章　OpenGL 中的图形变换

1、请使用 OpenGL、GLU 和 GLUT 编写一个三维犹他茶壶程序。其中茶壶的半径为 1 单位，并远移 3 单位；观察体规定为：视场角=60 度，宽高比=1，近=1，远=100；程序窗口的大小为(200, 200)，标题为“尤他茶壶”。

解:

```
#include <GL/glut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 100);
    glTranslatef(0, 0, - 3);
    glutSolidTeapot(1);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("尤他茶壶");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

2、请使用 OpenGL 和 GLUT 编写一个显示线框球体的简单图形程序。其中线框球体的半径为 0.8，经线数和纬线数均为 20，并绕 x 轴旋转 30 度，标题为“线框球”。

解:

```
#include <GL/glut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glRotated(30, 1, 0, 0);
    glutWireSphere(0.8, 20, 20);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("线框球");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

3、请使用 OpenGL、GLU 和 GLUT 编写一个三维犹他茶壶程序。其中茶壶的半径为 1 单位，并远移 3 单位；观察体规定为：视场角=60 度，宽高比=1，近=1，远=100；程序窗口的大小为(200, 200)，标题规定为“旋转的尤他茶壶”。茶壶绕 z 轴不断旋转，旋转的时间间隔为 50 毫秒，角度间隔为 5 度。注意旋转角度必须在 0 ~ 360 度以内。

解:

```
#include <gl/glut.h>
int angle = 0;
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 100);
    glTranslatef(0, 0, -3);
    glRotated(angle, 0, 0, 1);
```

```
    glutSolidTeapot(1);
    glFlush();
}
void timer(int millis)
{   angle = (angle + 5) % 360;
    glutPostRedisplay();
    glutTimerFunc(millis, timer, millis);
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("旋转的尤他茶壶");
    glutTimerFunc(50, timer, 50);
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

4、请使用 OpenGL、GLU 和 GLUT 编写一个简单的多视口演示程序。要求：在屏幕窗口左下角的 1/4 部分显示一个红色的填充正三角形；在屏幕窗口右上角的 1/4 部分显示一个绿色的填充正方形；三角形顶点坐标值的范围为 0～1，边长为 1；裁剪窗口为(-0.1, -0.1)～(1.1, 1.1)；程序窗口的大小为(200, 200)，标题为“多视口演示”。

解:

```
#include <gl/glut.h>
void Viewport(int x, int y, int w, int h)
{   glViewport(x, y, w, h);
    glLoadIdentity();
    gluOrtho2D(-0.1, 1.1, -0.1, 1.1);
}
void makeObject()
{   glBegin(GL_TRIANGLES);
    glVertex2d(0, 0);
    glVertex2d(1, 0);
    glVertex2d(0.5, 0.8660);
    glEnd();
}
void Paint()
{   int w = glutGet(GLUT_WINDOW_WIDTH) / 2;
    int h = glutGet(GLUT_WINDOW_HEIGHT) / 2;
    glClear(GL_COLOR_BUFFER_BIT);
    Viewport(0, 0, w, h);
    glColor3f(1, 0, 0);
    makeObject();
    Viewport(w, h, w, h);
    glColor3f(0, 1, 0);
    makeObject();
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("多视口演示");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

第 7 章　三维场景的真实感绘制

略

第 8 章　OpenGL 的真实感图形

1、使用 OpenGL 和 GLUT 编写一个程序，用于模拟一个非常光滑的球面在烈日暴晒下的效果。

解:

```
#include <gl/glut.h>
void Light()
{   float pos[] = { 2, 2, 2, 0 };
    glLightfv(GL_LIGHT0, GL_POSITION, pos);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
}
void Material()
{   float mater[] = {1, 1, 1, 1};
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mater);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mater);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 128);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 1000);
    glTranslatef(0, 0, -2);
    glutSolidSphere(0.95, 100, 50);
    glFlush();
}
int main()
{   glutInitWindowSize(300, 300);
    glutCreateWindow("光滑球面");
    glutDisplayFunc(Paint);
    Light(), Material();
    glutMainLoop();
}
```

2、在一个空旷的场景中有一个紫色的球，球的右上角方向放置了一个白色的点光源，请使用 OpenGL 和 GLUT 编写一个程序模拟出球面上的光照效果。

解:

```
#include <gl/glut.h>
void Light()
{   float pos[] = { 200, 200, 0, 1 };
    glLightfv(GL_LIGHT0, GL_POSITION, pos);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
}
void Material()
{   float mater[] = {1, 0, 1, 1};
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mater);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 1000);
    glTranslatef(0, 0, -2);
    glutSolidSphere(0.95, 100, 50);
    glFlush();
}
int main()
{   glutInitWindowSize(300, 300);
    glutCreateWindow("紫色的球");
    glutDisplayFunc(Paint);
    Light(), Material();
    glutMainLoop();
}
```

3、使用 OpenGL 和 GLUT 编写一个程序，绘制一个球体，在该球体上贴上一个渐变蓝色条纹的纹理。
解:
```
#include <gl/glut.h>
void TexImage()
{   GLubyte Texture[64][3];
    int i;
    for(i = 0; i < 64; i++)
        Texture[i][0] = Texture[i][1] = 255 - 2 * i, Texture[i][2] = 255;
    glTexImage1D(GL_TEXTURE_1D, 0, 3, 64, 0, GL_RGB, GL_UNSIGNED_BYTE, Texture);
}
void makeTex()
{   double params[] = { 1, 1, 1, 0 };
    TexImage();
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
    glTexGendv(GL_S, GL_OBJECT_PLANE, params);
    glEnable(GL_TEXTURE_1D);
    glEnable(GL_TEXTURE_GEN_S);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 1000);
    glTranslatef(0, 0, - 2);
    glutSolidSphere(1, 100, 50);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("渐变蓝色条纹的球面");
    glutDisplayFunc(Paint);
    makeTex();
    glutMainLoop();
}
```
4、在一个空旷的场景中由近到远有 3 个蓝色的球体，使用 OpenGL 和 GLUT 编写一个程序模拟出这 3 个球体在黄色雾中的效果。
解:
```
#include<gl/glut.h>
void Fog()
{   float fogColor[4] = { 1, 1, 0, 1 };
    glFogfv(GL_FOG_COLOR, fogColor);
    glFogi(GL_FOG_MODE, GL_LINEAR);
    glFogf(GL_FOG_DENSITY, 0.02);
    glFogf(GL_FOG_START, -2);
    glFogf(GL_FOG_END, 12);
    glHint(GL_FOG_HINT, GL_DONT_CARE);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}
void Material()
{   float color[] = {0, 0, 1, 1};
    glMaterialfv(GL_FRONT, GL_DIFFUSE, color);
}
void MakeObj()
{   int i;
```

```
    for(i = 0; i < 3; i++)
    {   glutSolidSphere(0.8, 100, 50);
        glTranslatef(1.25, 1, -1.5);
    }
}
void Reshape(int w, int h)
{   glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 1000);
    glMatrixMode(GL_MODELVIEW);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(-0.5, -0.5, -3);
    MakeObj();
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("雾中的球");
    glutReshapeFunc(Reshape);
    glutDisplayFunc(Paint);
    Fog(), Material(), glEnable(GL_FOG);
    glutMainLoop();
}
```

第 9 章　样条方法和分形几何方法

1、请指出插值样条和逼近样条的区别。

2、假设在控制点 p_k 和 p_{k+1} 之间的曲线段是参数三次函数 $p(u)$，Hermite 曲线段的边界条件是什么？请解释所使用符号的含义。

3、请写出 Bézier 样条曲线基函数的定义。

4、请写出 B-样条曲线基函数的定义。

5、使用 Hermite 方法求 $P_0(0,0)$ 和 $P_1(5,6)$ 之间的曲线段 $p(u)$（$0\le u\le 1$），并计算参数值为 0.5 时的结果，其中 $p(0)=(0,0)$，$p(1)=(5,6)$，$p'(0)=(3,8)$，$p'(1)=(5,1)$。

解:

$$p(u)=(u^3\ u^2\ u\ 1)\begin{pmatrix}2&-2&1&1\\-3&3&-2&-1\\0&0&1&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}0&0\\5&6\\3&8\\5&1\end{pmatrix}=(-2u^3+4u^2+3u,-3u^3+u^2+8u)$$

$$p(0.5)=(-2\times0.5^3+4\times0.5^2+3\times0.5,-3\times0.5^3+0.5^2+8\times0.5)=(2.25,3.875)$$

6、给定四个控制点 $P_0(0,0,0)$、$P_1(1,1,1)$、$P_2(2,-1,-1)$ 和 $P_3(3,0,0)$，请构造一条三次 Bézier 曲线，并计算参数为 0、1/3、2/3 和 1 时的值。

解:

$$p(u)=\begin{pmatrix}u^3&u^2&u&1\end{pmatrix}\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}0&0&0\\1&0&1\\2&0&0\\3&0&0\end{pmatrix}=\begin{pmatrix}u^3&u^2&u&1\end{pmatrix}\begin{pmatrix}0&0&3\\0&0&-6\\3&0&3\\0&0&0\end{pmatrix}=\begin{pmatrix}3u&0&3u^3-6u^2+3u\end{pmatrix}$$

所以，$p(0)=(0,0,0)$，$p(1/3)=(1,0,4/9)$，$p(2/3)=(2,0,2/9)$，$p(1)=(3,0,0)$。

7、给定三个控制点 $P_0(0,0,0)$、$P_1(50,60,0)$ 和 $P_2(100,10,0)$，请构造一条二次均匀 B-样条曲线。

解:

取 $n=2$，$d=3$，节点向量含有 $n+d+1=6$ 个节点值：$\{0,1,2,3,4,5\}$。

首先计算第一个基函数: $N_{03}(u)$, $0\le u<3$。

（1）当 $0\le u<1$ 时， $N_{12}(u)=0$, $N_{11}(u)=0$, $N_{01}(u)=1$，

$$N_{02}(u)=\frac{u-u_0}{u_1-u_0}N_{01}(u)=u$$
$$N_{03}(u)=\frac{u-u_0}{u_2-u_0}N_{02}(u)=\frac{1}{2}u^2$$

（2）当 $1\le u<2$ 时，由周期性，有 $N_{12}(u)=u-1$，

$$N_{11}(u)=1,\ N_{01}(u)=0,$$
$$N_{02}(u)=\frac{u_2-u}{u_2-u_1}N_{11}(u)=2-u$$
$$N_{03}(u)=\frac{u-u_0}{u_2-u_0}N_{02}(u)+\frac{u_3-u}{u_3-u_1}N_{12}(u)=\frac{1}{2}u(2-u)+\frac{1}{2}(u-1)(3-u)$$

（3）当 $2\le u<3$ 时，由周期性，有 $N_{12}(u)=3-u$, $N_{02}(u)=0$，

$$N_{03}(u)=\frac{u_3-u}{u_3-u_1}N_{12}(u)=\frac{1}{2}(3-u)^2$$

整合上述结果，得到

$$N_{13}(u)=\begin{cases}\frac{1}{2}u^2 & 0\le u<1\\ \frac{1}{2}u(2-u)+\frac{1}{2}(u-1)(3-u) & 1\le u<2\\ \frac{1}{2}(3-u)^2 & 2\le u<3\end{cases}$$

然后由周期性计算其余两个基函数。

$$N_{13}(u)=\begin{cases}\frac{1}{2}(u-1)^2 & 1\le u<2\\ \frac{1}{2}(u-1)(3-u)+\frac{1}{2}(u-2)(4-u) & 2\le u<3\\ \frac{1}{2}(4-u)^2 & 3\le u<4\end{cases}$$

$$N_{23}(u)=\begin{cases}\frac{1}{2}(u-2)^2 & 2\le u<3\\ \frac{1}{2}(u-2)(4-u)+\frac{1}{2}(u-3)(5-u) & 3\le u<4\\ \frac{1}{2}(5-u)^2 & 4\le u<5\end{cases}$$

多项式曲线 $p(u)$ 的参数范围： $u_{d-1}\le u\le u_{n+1}$，即 $2\le u\le 3$，考虑到多项式曲线 $p(u)$ 具有一阶参数连续性，可以如下定义 $p(u)$：

当 $2\le u\le 3$ 时，

$$\begin{aligned}p(u)&=P_0\times\frac{1}{2}(3-u)^2+P_1\times\left[\frac{1}{2}(u-1)(3-u)+\frac{1}{2}(u-2)(4-u)\right]\\&\quad+P_2\times\frac{1}{2}(u-2)^2\\&=\frac{1}{2}(u^2-6u+9)\times(0,0,0)+\frac{1}{2}(-2u^2+10u-11)\times(50,60,0)\\&\quad+\frac{1}{2}(u^2-4u+4)\times(100,10,0)\\&=\frac{1}{2}(100u-150,-11u^2+560u-620,0)\end{aligned}$$

8、给定四个控制点 $P_0(0,0,0)$、$P_1(1,0,1)$、$P_2(2,0,0)$ 和 $P_3(3,0,0)$，请使用边界条件定义的方法构造一条三次均匀 B-样条曲线，并计算参数为 0，1/3，2/3，1 时的值。

解:

$$p(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\frac{1}{6}\begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{pmatrix}\begin{pmatrix}0 & 0 & 0\\ 1 & 0 & 1\\ 2 & 0 & 0\\ 3 & 0 & 0\end{pmatrix}=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\frac{1}{6}\begin{pmatrix}0 & 0 & 3\\ 0 & 0 & -6\\ 6 & 0 & 0\\ 6 & 0 & 4\end{pmatrix}=\frac{1}{6}\begin{pmatrix}6u+6 & 0 & 3u^3-6u^2+4\end{pmatrix}$$

所以， $p(0)=\frac{1}{6}(6,0,4)$， $p(1/3)=\frac{1}{6}(8,0,31/9)$， $p(2/3)=\frac{1}{6}(10,0,20/9)$， $p(1)=\frac{1}{6}(12,0,1)$。

第 10 章　OpenGL 的样条和分形**

1、使用随机迭代算法和下表中的参数编写 3 个程序，分别绘制出分形蕨、分形草和分形树的图像。

第 1 题

参数	*a*	*b*	*e*	*c*	*d*	*f*	*p*
蕨	0	0	0	0	0.16	0	0.10
	0.2	-0.26	0	0.23	0.22	1.6	0.08
	-0.15	0.28	0	0.26	0.24	0.44	0.08
	0.75	0.04	0	-0.04	0.85	1.6	0.74
草	0	0	0	0	0.5	0	0.15
	0.02	-0.28	0	0.15	0.2	1.5	0.10
	0.02	0.28	0	0.15	0.2	1.5	0.10
	0.75	0	0	0	0.5	4.6	0.65
树	0.05	0	0	0	0.6	0	0.1
	0.05	0	0	0	-0.5	1	0.1
	0.46	-0.321	0	0.386	0.383	0.6	0.2
	0.47	-0.154	0	0.171	0.423	1.1	0.2
	0.433	0.275	0	-0.25	0.476	1	0.2
	0.421	0.257	0	-0.353	0.306	0.7	0.2

解：参阅 10.7.2 和 10.7.3 小节，修改相应数据即可。

2、使用 OpenGL 及相关程序库编写一个程序，绘制 Mandelbrot 集合在复数窗口 $(-1,0) \sim (-0.5,0.5)$ 内的图像。

解:

```
#include <GL/glut.h>
#include <complex>
using namespace std;
typedef complex<double> Complex;
Complex C1(-1, 0), C2(-0.5, 0.5);
void Color(GLubyte x)
{
#define Bit(val, num) (((val)>>(num))&1)
    GLubyte r, g, b, s;
    s = Bit(x, 7) << 4 | Bit(x, 3) << 5;
    r = Bit(x, 6) << 6 | Bit(x, 2) << 7 | s;
    g = Bit(x, 5) << 6 | Bit(x, 1) << 7 | s;
    b = Bit(x, 4) << 6 | Bit(x, 0) << 7 | s;
    glColor3ub(r, g, b);
#undef Bit
}
int iterate(Complex z, Complex C, int maxIter)
{   int count;
    for(count = 0; norm(z) <= 4 && count < maxIter; count++)
        z = z * z + C;
    return count;
}
void mandelbrot(int nx, int ny, int maxIter)
{   double x, dx = (C2.real() - C1.real()) / nx;
    double y, dy = (C2.imag() - C1.imag()) / ny;
    glBegin(GL_POINTS);
    for(x = C1.real(); x < C2.real(); x += dx)
        for(y = C1.imag(); y < C2.imag(); y += dy)
        {   Color(iterate(Complex(x, y), Complex(x, y), maxIter));
            glVertex2d(x, y);
        }
    glEnd();
}
void Reshape(int w, int h)
```

```
{   glViewport(0, 0, w, h);
    glLoadIdentity();
    gluOrtho2D(C1.real(), C2.real(), C1.imag(), C2.imag());
}
void Paint()
{   const int maxIter = 256;
    int w = glutGet(GLUT_WINDOW_WIDTH);
    int h = glutGet(GLUT_WINDOW_HEIGHT);
    glClear(GL_COLOR_BUFFER_BIT);
    mandelbrot(w + 1, h + 1, maxIter);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("Mandelbrot Set");
    glutDisplayFunc(Paint);
    glutReshapeFunc(Reshape);
    glutMainLoop();
}
```

3、使用 OpenGL 及相关程序库编写一个程序，绘制当 $C = -0.5 + i$ 时 Julia 集合在复数窗口 $(-1, 0) \sim (-0.5, 0.5)$ 内的图像。

解:

```
#include <GL/glut.h>
#include <complex>
using namespace std;
typedef complex<double> Complex;
Complex C1(-1, 0), C2(-0.5, 0.5);
void Color(GLubyte x)
{
#define Bit(val, num) (((val)>>(num))&1)
    GLubyte r, g, b, s;
    s = Bit(x, 7) << 4 | Bit(x, 3) << 5;
    r = Bit(x, 6) << 6 | Bit(x, 2) << 7 | s;
    g = Bit(x, 5) << 6 | Bit(x, 1) << 7 | s;
    b = Bit(x, 4) << 6 | Bit(x, 0) << 7 | s;
    glColor3ub(r, g, b);
#undef Bit
}
int iterate(Complex z, Complex C, int maxIter)
{   int count;
    for(count = 0; norm(z) <= 4 && count < maxIter; count++)
        z = z * z + C;
    return count;
}
void Julia(Complex C, int nx, int ny, int maxIter)
{   double x, dx = (C2.real() - C1.real()) / nx;
    double y, dy = (C2.imag() - C1.imag()) / ny;
    glBegin(GL_POINTS);
    for(x = C1.real(); x < C2.real(); x += dx)
        for(y = C1.imag(); y < C2.imag(); y += dy)
        {   Color(iterate(Complex(x, y), C, maxIter));
            glVertex2d(x, y);
        }
    glEnd();
}
void Reshape(int w, int h)
{   glViewport(0, 0, w, h);
    glLoadIdentity();
    gluOrtho2D(C1.real(), C2.real(), C1.imag(), C2.imag());
```

```
}
void Paint()
{   const int maxIter = 256;
    int w = glutGet(GLUT_WINDOW_WIDTH);
    int h = glutGet(GLUT_WINDOW_HEIGHT);
    glClear(GL_COLOR_BUFFER_BIT);
    Julia(Complex(-0.5, 1), w + 1, h + 1, maxIter);
    glFlush();
}
int main()
{   glutInitWindowSize(200, 200);
    glutCreateWindow("Mandelbrot Set");
    glutDisplayFunc(Paint);
    glutReshapeFunc(Reshape);
    glutMainLoop();
}
```

4、给定四个控制点 $P_0(0,0,0)$、$P_1(1,1,1)$、$P_2(2,-1,-1)$和$P_3(3,0,0)$，编写 1 个程序绘制这些控制点生成的三次 Bézier 曲线。

解:

```
#include <gl/glut.h>
void Reshape(int w, int h)
{   glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, (double)w / h, 1, 1000);
    glTranslatef(-1.5, 0, -2);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void BezCurve(float *pts, int Npts, int un)
{   glMap1f(GL_MAP1_VERTEX_3, 0, 1, 3, Npts, pts);
    glEnable(GL_MAP1_VERTEX_3);
    glMapGrid1f(un, 0, 1);
    glEvalMesh1(GL_LINE, 0, un);
    glDisable(GL_MAP1_VERTEX_3);
}
void Paint()
{   enum {Npts = 4};
    float pts[Npts][3] = {{0, 0, 0}, {1, 1, 1}, {2, -1, -1}, {3, 0, 0}};
    int un = 50;
    glClear(GL_COLOR_BUFFER_BIT);
    BezCurve(*pts, Npts, un);
    glFlush();
}
int main()
{   glutInitWindowSize(400, 300);
    glutCreateWindow("Bézier 曲线");
    glutReshapeFunc(Reshape);
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```