# Asymptote

```
/*
size(200);
path my_path0 = (0,0)--(1,1);
path my_path1 = (0,0)--(100,-100);
draw(my_path0, arrow = Arrow);
draw((0.5,0.5),my_path1, arrow = Arrow, bar = Bar);
*/

/*
//围绕曲面的向量矢量场 FSS天线罩
import graph3;
size(200);
triple f(pair z){return expi(z.x,z.y);}
path3 vector(pair z) {
real t=sqrtEpsilon;
triple U=(expi(z.x,z.y+t)-expi(z.x,z.y))/t;
U=unit(U);
return O--U;
}
add(vectorfield(vector,f,(0,0),(pi,2pi),40, maxlength=0.015,yellow+3+opacity(0.5), arrow = None));
draw(unitsphere,palegray);
limits((-1.5,-1.5,-1.5),(1.5,1.5,1.5));
axes3("𝒙","𝒚","𝒛",red,Arrow3());
*/

/*
//画空洞的物体 主要用了planar=true选项，而且要用reverse调顺逆时针路径
size(200);
import three;
currentprojection=orthographic(3,4,2);
path p=reverse(polygon(6));
path q=scale(0.3)*unitcircle;
path3 pp=path3(p);
path3 qq=path3(q);
draw(surface(pp^^qq,planar=true),red);
draw(surface(shift(Z)*pp^^qq,planar=true),blue);
for(path h : p)
draw(extrude(h,Z),green+opacity(0.5),black+1pt);
for(path h : q)
draw(extrude(h,Z),green);
*/

/*
//extrude函数画直纹面 画柱面和圆锥
import three;
size(200);
path3 p=shift(Z)*unitcircle3;
path3 q=scale3(2)*unitcircle3;
draw(p^^q,yellow+1bp);
//传入两个path3，会自动填补之间的距离
draw(extrude(p,q),magenta+opacity(0.25));
*/
```

```
/*无限循环动画....
settings.loop=true;
settings.autoplay=true;
import graph3;
import animate;
currentprojection=orthographic(1,-2,0.5);
animation A;
int n=25;
for(int i=0; i < n; ++i) {
picture pic;
size3(pic,6cm);
real k=i/n*pi;
real f(pair z) {return 4cos(abs(z)-k)*exp(-abs(z)/6);}
draw(pic,surface(f,(-4pi,-4pi),(4pi,4pi),Spline),paleblue);
draw(pic,shift(i*6Z/n)*unitsphere,yellow);
A.add(pic);
}
A.glmovie();
*/

/*
//3d动画 pdf多图
settings.prc=false;
settings.render=4;
import graph3;
import animation;
viewportmargin=(10,10);
currentprojection=orthographic((19,11,18));
currentlight=(0,5,5);
unitsize(1cm);
real R=3;
real a=1;
int n=4;
triple f(pair t) {
return ((R+a*cos(t.y))*cos(t.x),(R+a*cos(t.y))*sin(t.x),a*sin(t.y));
}
surface s=surface(f,(0,0),(2pi,2pi),8,8,Spline);
typedef triple function(real);
function g(int k){
return new triple(real s){
return ((R-a*cos(2*pi*s))*cos(2*pi/(1+k+s)),
(R-a*cos(2*pi*s))*sin(2*pi/(1+k+s)),
-a*sin(2*pi*s));
};
}
animation A;
picture fig;
size3(fig,200);
for(int i=0; i < n; ++i){
draw(fig,s,yellow);
A.add(fig);
for(int j=0;j <= i; ++j){
draw(fig,graph(g(j),0,1,operator..),blue+linewidth(2));
}
```

```
A.add(fig);
}
A.movie(delay=400);
*/

/*
//动画
import animate;
pair[] P={(5,0),(-8,50),(55,50),(100,0)};
defaultpen(fontsize(20pt)+1.5pt);
animation Ani;
int n=30;
for(int i=0;i picture pic;
size(pic,400);
real t=i/n;
pair A=interp(P[0],P[1],t);
pair B=interp(P[1],P[2],t);
pair C=interp(P[2],P[3],t);
pair U=interp(A,B,t);
pair V=interp(B,C,t);
pair W=interp(U,V,t);
draw(pic,P[0]--P[1]--P[2]--P[3]);
draw(pic,A--B--C,green);
draw(pic,U--V,blue);
draw(pic,P[0]..controls A and U.. W,red);
Ani.add(pic);
}
Ani.pdf(delay=400,"control");
*/

/*
//mobius带原理
import graph3;
size(200);
real R=6;
real r=1;
triple Mobius(pair w){
real t=w.x;
real l=w.y;
return R*expi(pi/2,t)+l*expi(t/2,t);
}
surface s=surface(Mobius,(0,-r),(2pi,r),30,Spline);
draw(Circle(O,R,Z));
draw(s,lightgreen+opacity(0.6));
for(real c: new real[]{0,0.05pi,0.1pi,0.15pi,0.3pi,0.5pi,0.8pi,pi,1.4pi,1.6pi,1.9pi}){
triple Center=R*expi(pi/2,c);
triple U=r*expi(0,c);
triple V=r*expi(c/2,c);
triple A=Center+U;
triple B=Center+V;
draw(Center--A,red);
draw(Center--B,red);
draw(O--(R+r)*expi(pi/2,c));
if(c!=0){
```

```
draw(arc(Center,A,B),red,Arrow3(1mm));
draw(surface(Center--A--arc(Center,A,B)--Center--cycle),cyan,nolight);
draw(surface(Center--A--arc(Center,A,B,CW)--Center--cycle),yellow,nolight);
}
}
zaxis3(Arrow3());
*/

/*
//双曲柱面，椭圆柱面，抛物柱面
//以下例子是抛物柱面，主要使用了颜色以及z对称的方法
import contour;
import palette;
import graph3;
currentprojection=orthographic(5,10,10);
limits((-4,-12,-6),(14,12,6));
size(0,200);
real p=3;
real f(real x, real y){return y^2-2p*x;}
guide[][] g=contour(f,(-10,-10),(10,10),new real[]{1},50);
for(path p:g[0]){
surface sf=extrude(p,4Z);
sf.colors(palette(sf.map(xpart),Rainbow()));
draw(surface(sf,zscale3(-1)*sf));
draw(path3(p),red+2pt);
}
axes3("𝒙","𝒚","𝒛",Arrow3());
*/

/*
//双曲抛物面
import graph3;
import contour3;
import palette;
real f(real x, real y, real z){
return x^2-y^2-z;
}
size(10cm);
currentprojection=orthographic(2.5,5,5);
surface sf=surface(contour3(f,(-5,-5,-5),(5,5,5),15));
sf.colors(palette(sf.map(zpart),Rainbow()));
draw(sf,nolight);
axes3("𝒙", "𝒚", "𝒛", Arrow3());
*/

/*
//椭圆抛物面
import graph3;
import palette;
size(0,200);
currentprojection=perspective(10,10,10);
triple f(pair t) {
real a=4;
real b=3;
```

```
real c=5;
real u=t.x;
real v=t.y;
return (a*v*cos(u),b*v*sin(u),c*v^2);
}
surface s=surface(f,(0,0),(2pi,1),40,40,Spline);
s.colors(palette(s.map(zpart),Rainbow()));
draw(s);
axes3("x","y","z",Arrow3());
*/

/*
//椭球面
import graph3;
import palette;
size(200,0);
currentprojection=perspective(10,10,10);
limits((-6,-5,-4),(6,5,4));
triple f(pair t){
real a=5;
real b=4;
real c=3;
real r=t.x;
real theta=t.y;
real x=a*r*cos(theta);
real y=b*r*sin(theta);
//real z=sqrt(c^2*(1-x^2/a^2+y^2/b^2));
real z=sqrt(1-r^2)*c;
return (x,y,z);
}
surface s=surface(f,(0,0),(1,2pi),40,40,Spline);
s.colors(palette(s.map(zpart),Rainbow()));
draw(surface(s,zscale3(-1)*s));
axes3("x","y","z",Arrow3());
*/

/*
//双叶双曲面
import graph3;
import palette;
size(0,200);
currentprojection=perspective(10,10,10);
limits((-10,-10,-20),(10,10,20));
triple f(pair t){
real a=4;
real b=3;
real c=5;
real r=t.x;
real theta=t.y;
real x=a*r*cos(theta);
real y=b*r*sin(theta);
//real z=sqrt(c^2*(x^2/a^2+y^2/b^2+1));
real z=sqrt(r^2+1)*c;
return (x,y,z);
```

```
}
surface s=surface(f,(0,0),(5,2pi),40,40,Spline);
s.colors(palette(s.map(zpart),Rainbow()));
draw(surface(s,zscale3(-1)*s));
axes3("x","y","z",Arrow3());
*/

/*
//单叶双曲面,用了椭圆坐标系
import graph3;
import palette;
size(0,200);
currentprojection=perspective(10,10,10);
limits((-10,-10,-20),(10,10,20));
triple f(pair t) {
real a=4;
real b=3;
real c=5;
real r=t.x;
real theta=t.y;
real x=a*r*cos(theta);
real y=b*r*sin(theta);
//real z=sqrt(c^2*(x^2/a^2+y^2/b^2-1));
real z=sqrt(r^2-1)*c;
return (x,y,z);
}
surface s=surface(f,(1,0),(3,2pi),50,Spline);
s.colors(palette(s.map(abs),Wheel()));
//主要是sqrt只返回正数，所以需要zscale3(-1)进行翻转
draw(surface(s,zscale3(-1)*s));
axes3("x","y","z",Arrow3());
*/

/*
//contour3例子
import graph3;
import contour3;
import palette;
size(200);
currentprojection=orthographic(6,8,2);
limits((-22,-22,-22),(22,22,22));
real c0=0.1;
//直角坐标转为球坐标系
real h(real x,real y,real z){
triple v=(x,y,z);
real rho=length(v);
real theta=polar(v,warn=false);
real phi=azimuth(v,warn=false);
return rho*(1-rho/6)*exp(-rho/3)*cos(theta)-c0;
}
surface s=surface(contour3(h,(-20,-20,-20),(20,20,20),30));
s.colors(palette(s.map(abs),Gradient(palegreen,heavyblue)));
draw(s);
draw(zscale3(-1)*s);
```

```
axes3("$x$","$y$","$z$",Arrow3());
*/

/*
//contour3画更复杂的隐函数曲面
import graph3;
import palette;
import contour3;
size(200);
currentprojection=perspective(10,8,8);
real f(real x, real y, real z){
return cos(x)*sin(y)+cos(y)*sin(z)+cos(z)*sin(x);
}
surface sf=surface(contour3(f,(-2pi,-2pi,-2pi),(2pi,2pi,2pi),12));
sf.colors(palette(sf.map(abs),Gradient(red,yellow)));
draw(sf,nolight);
xaxis3("$x$",Bounds(),blue,InTicks());
yaxis3("$y$",Bounds(),blue,InTicks());
zaxis3("$z$",Bounds(),blue,InTicks());
*/

/*
//contour3函数
//f(x,y,z)=x^2+y^2+z^2-4相当于在0处的等值面
import graph3;
import contour3;
real f(real x, real y,real z)
{
return x^2+y^2+z^2-4;
}
vertex[][] vs = contour3(f,(-2.1,-2.1,-2.1),(2.1,2.1,2.1),15);
draw(surface(vs),blue);
*/

/*
//返回空间曲面和曲线的相交点,intersectionpoints(path3, patch or surface);
size(200);
import three;
path3 g=randompath3(15);
draw(g,red+thin());
surface s=unitsphere;
draw(s,lightyellow);
dot(intersectionpoints(g,s),red);
*/

/*
//math宏包提供了直线段和平面交点的intersect函数,返回t=0~1的值，还需要再在线段上取实际值
import math;
import graph3;
size(200);
currentprojection=orthographic(2.5,6.1,1.4);
path3[] P={
X--Y--Z--cycle,
Y--(-X)--Z--cycle,
(-X)--(-Y)--Z--cycle,
```

```
(-Y)--X--Z--cycle
};
for(int i=0;i draw(surface(P[i]),lightred);
draw(surface(zscale3(-1)*P[i]),lightred);
draw((P[i]));
draw(zscale3(-1)*P[i]);
}
triple A=interp(-Y,Z,0.3);
triple B=interp(X,Z,0.5);
triple K=-Z+0.2X;
triple n=cross(A-K,B-K);
real tC=intersect(X,Y,n,K);
triple C=point(X--Y,tC);
real tD=intersect(X,-Z,n,K);
triple D=point(X--(-Z),tD);
real tE=intersect(-Y,-Z,n,K);
triple E=point((-Y)--(-Z),tE);
real tF=intersect((-Y),(-X),n,K);
triple F=point((-Y)--(-X),tF);
path3 p=A--B--C--D--E--F--cycle; draw(p,red+1pt);
draw(surface(p--cycle),green);

limits((-1.2,-1.2,-1.2),(1.2,1.2,1.2));
axes3("𝒙","𝒚","𝒛",Arrow3());
*/

/*
//牟合方盖，两个圆柱相交所成的立体，还是利用极坐标系的例子
size(200);
import graph3;
currentprojection=orthographic(4,2,0);
real a=2;
//这条方程已经是两个柱体相减后的了
real f(real x,real y){
return sqrt(a^2-x^2);
}
triple g(pair t){
real r=t.x;
real theta=t.y;
return (r*cos(theta),r*sin(theta),f(r*cos(theta),r*sin(theta)));
}

surface S=surface(g,(0,0),(a,2pi),50);
draw(S,brown);
draw(zscale3(-1)*S,brown);
draw(rotate(90,X)*S,springgreen);
draw(rotate(-90,X)*S,springgreen);

limits((-1.5a,-1.5a,-1.5a),(1.5a,1.5a,1.5a));
xaxis3("𝒙",Arrow3());
yaxis3("𝒚",Arrow3());
zaxis3("𝒛",Arrow3());
*/

/*
```

```
//如何利用椭圆坐标系的定义域截断曲线的外围部分，画出有两个不等式的曲面
import contour;
import graph3;
size(0,200);
currentprojection=orthographic(50,50,50);
limits((-3,-2,0),(3,2,8));
real f(pair t){return 8-t.x^2-t.y^2;}
real g(pair t){return t.x^2+3t.y^2;}
real d(pair t) {
return t.x^2/4+t.y^2/2-1;
// return f(t)-g(t);
}
guide[][] conto = contour(d, (-3,-3), (3,3), new real[]{0});
draw(surface(conto[0][0]..cycle),palegray); //其实这里不严谨，因为有时候我们并知不道path[]数组是否只有一个数据
draw(path3(conto[0][0]),red+dashed);
draw(lift(g,conto),red+1pt);
for(guide p: conto[0]){
draw(extrude(p,8Z),palegray+opacity(0.1)); //从底面拉出一个曲面
}

triple F(pair t){
real r=t.x;
real theta=t.y;
real x=2*r*cos(theta);
real y=sqrt(2)*r*sin(theta);
real z=8-x^2-y^2;
return (x,y,z);}

triple G(pair t) {
real r=t.x;
real theta=t.y;
real x=2*r*cos(theta);
real y=sqrt(2)*r*sin(theta);
real z=x^2+3y^2;
return (x,y,z);}

//这个椭圆定义域刚好把相交线外面的区域全部截断了
draw(surface(F,(0,0),(1,2pi),10,Spline),paleyellow);
draw(surface(G,(0,0),(1,2pi),10,Spline),palegreen);
xaxis3("𝒙",Arrow3);
yaxis3("𝒚",Arrow3);
*/

/*
//转换为极坐标的另一个例子Enneper-曲面
import graph3;
size(200,0);
currentprojection=perspective(50,-200,50);
triple f(pair t) {
real r=t.x;
real theta=t.y;
pair w=(r*cos(theta),r*sin(theta));
real x=xpart(w-w*w*w/3);
real y=xpart((0,1)*(w+w*w*w/3));
real z=xpart(w*w);
```

```
return (x,y,z);
}
draw(surface(f,(0,0),(1.6,2pi),60,40,Spline),nullpen,meshpen=currentpen);
*/

/*
//当前例子的下一个例子画出来的效果不好，进行极坐标表换后的例子
import graph3;
size(200,0);
currentprojection=orthographic(3,2,4);
real f(real x,real y){return x^2-y^2;}
// 对f进行极坐标变换;
triple g(pair t){
real r=t.x;
real theta=t.y;
real x=r*cos(theta);
real y=r*sin(theta);
real z=f(x,y);
return (x,y,z);
}
draw(surface(g,(0,0),(1,2pi),20,Spline),red+opacity(0.5),black);
*/

/*
//画曲面的局部
import graph3;
size(200);
currentprojection=orthographic(3,2,4);
real f(pair z){return z.x^2-z.y^2;}

//主要是在根据方程构造surface曲面的时候加入一个返回值为bool,传入值为pair的筛选函数
draw(surface(f,(-1.2,-1.2),(1.2,1.2),50,Spline,
new bool(pair z){return z.x^2+z.y^2 <=1;}),red+opacity(0.5),black);
draw(surface(f,(-1.2,-1.2),(1.2,1.2),50,Spline,
new bool(pair z){return z.x^2+z.y^2 >1;}),blue+opacity(0.5),black);
*/

/*
//层次分明的等高线曲线图绘制
import graph3;
import contour;
import palette;
size(8cm,7cm,IgnoreAspect);
currentprojection=orthographic(-10,-5,4);
real f(pair z) {return (z.x+z.y)/(2+cos(z.x)*sin(z.y));}
real[] levels={2,4,6,8};
surface s=surface(f,(0,0),(5,10),100);
//根绝find返回的序号填充颜色;
s.colors(palette(s.map(new real(triple v) {return find(levels > v.z);}),Rainbow()));
draw(s);
draw(lift(f,contour(f,(0,0),(5,10),levels)),2bp+red);
*/

/*
//等高线投影
```

```
import graph3;
import contour;
import palette;
size(8cm,7cm,IgnoreAspect);
currentprojection=orthographic(0,0,4);
real f(pair z) {return (z.x+z.y)/(2+cos(z.x)*sin(z.y));}
real[] levels={2,4,6,8};
surface s=surface(f,(0,0),(5,10),20,Spline);
s.colors(palette(s.map(zpart),Rainbow()));
//投影,这里planeproject缺省值取了XY平面
draw(planeproject(unitsquare3)*s,nolight);
guide[][] pl=contour(f,(0,0),(5,10),levels);
for(int i=0;i for(int j=0;j draw(path3(pl[i][j]),blue);
}
}
*/

/*
import three;
import graph3;
import contour;
import palette;
size(200);
currentprojection=orthographic(-10,-5,4);
real f(pair z) {return (z.x+z.y)/(2+cos(z.x)*sin(z.y));}
real[] levels={2,4,6,8};
guide[][] gs = contour(f,(0,0),(5,10),levels,30);
draw(lift(f, gs), red);
surface s = surface(f,(0,0),(5,10),Spline);
s.colors(palette(s.map(zpart), Rainbow()+opacity(0.5)));
draw(s);

xaxis3(Label("𝒙",position=MidPoint,align=SE),Bounds(Min,Min),OutTicks());
yaxis3(Label("𝒚",position=MidPoint,align=SW),Bounds(Min,Min),OutTicks(Step=2));
zaxis3(Label("𝒛
",position=EndPoint,align=N+W),XYEquals(0,10),InTicks(beginlabel=false,endlabel=false,Label(align=Y)));

//最主要的还是把guide[][]转换为path3再进行单独处理，否则会当成二维图纸画
for(int i=0; i < gs.length; ++i)
{
for(int j=0; j {
draw(path3(gs[i][j]), blue);
}
}
*/

/*
//这里展示了如何向等高线中添加注释文字等
import graph3;
import contour;
size(7.5cm,0);
size3(7.5cm,IgnoreAspect);
currentprojection=orthographic(-10,-10,200);
real f(pair z)
{
```

```
return 2z.x^2-z.x+z.y^2;
}
real[] values = {50,100,150,200,200,250};
guide[][] gs = contour(f, (0,0), (10,10), values);
draw(surface(f, (0,0), (10,10)), surfacepen=blue+opacity(0.5), meshpen=black);

//这里展示了怎么在等高线处添加文字标签注释
Label[] L = sequence(
new Label(int i)
{
return Label(format("𝑧＝", values[i]), align=E, MidPoint);
}
, values.length);
draw(L ,lift(f, gs), red);
*/

/*
//画等高线,其实还是用到了lift函数
import contour;
import palette;
import graph3;
currentprojection = orthographic(0,0,25);
size(0,12cm);
real a=3;
real b=4;
real f(pair z){return (z.x+z.y)/(2+cos(z.x)*sin(z.y));}
guide[][] g=contour(f,(-10,-10),(10,10), new real[]{8},150);
//这里把g[0]的每一小段path都进行了处理
for(guide p:g[0])
{
draw(extrude(p,8Z),palered);
draw(path3(p), green + 2pt);
}
draw(lift(f,g), red+2pt);
surface s=surface(f,(0,0),(10,10),20,Spline);
s.colors(palette(s.map(zpart), Rainbow()+opacity(0.9)));
draw(s);
*/

/*
//把字体贴到表面上
settings.tex="xelatex";
settings.prc=false;
texpreamble("\usepackage{xeCJK}");
texpreamble("\setCJKmainfont{SimSun}");
import graph3 ;
size(200);
currentprojection=orthographic(39.4,-264.6,121) ;
real f(pair z) {return 0.5+exp(-abs(z)^2);}
draw(surface(f,(-2.5,-2.5),(2.5,2.5),20,Spline),lightgray);
//关键是下面这一段的处理，把path[]转换为guide[][];
path[] china=scale(0.2)*texpath("中国");
guide[][] gui;
for(int i=0;i {
```

```
gui[i]=new guide[];
int n=8;
for(int j=0;j {
gui[i][j]=subpath(china[i],j/n,(j+1)/n);
}
}

draw(lift(f,gui),yellow+1pt);
*/

/*
//二维曲线映射到三维的曲面中
import graph3;
import contour;
size(200);
currentprojection = orthographic(2,3,6,up=Z);

real g(pair z){return z.x^2-z.y^2;}
draw(surface(g,(-3,-3),(3,3),Spline),green+opacity(0.5));
real f(pair z)
{
real x=z.x;
real y=z.y;
return x^2+y^2;
}
guide[][] gui=contour(f,(-2,-2),(2,2),new real[] {1,3});
//画出的曲线直接升到g曲面上
draw(lift(g,gui),red+linewidth(2));
axes3("𝒙","𝒚","𝒛",opacity(0.2),Arrow3);
*/

/*
import graph3;
import contour;
size(200,IgnoreAspect);
currentprojection=perspective(-2,-2,2);
triple f(pair t){
real x=t.x;
real y=x^2;
real z=t.y;
return (x,y,z);
}
real g(pair t){
real x=t.x;
real y=t.y;
return x^3;
}
draw(surface(f,(-2,-3),(2,3),Spline),red);
draw(surface(g,(-2,-3),(2,3),Spline),blue);
//contour接收二元参数，返回real值。
real h(pair t){
real x=t.x;
real y=t.y;
return y-x^2;
}
```

```
draw(lift(g,contour(h,(-2,-2),(2,2),new real[]{0})),linewidth(4bp)+yellow);
*/

/*
//曲面上画曲线，曲线方程不知，借助lift函数
//所画的锥体落在由z = g(x, y) = sqrt(x^2+y^2)/c 确定的曲面上,
//而球的方程是(x-Os.x)^2+(y-Os.y)^2+(z-Os.z)^2-r^2 = 0, 从这两个方程中消去z,
//得到的就是(x-Os.x)^2+(y-Os.y)^2+(g(x, y)-Os.z)^2-r^2 = 0,
//这个正是两个曲面交线在XY平面的投影方程. 我们用contour生成该曲线,
//然后再用lift函数把这个曲线提升到g(x,y) 上, 由此即画出两个曲面交线.
import graph3;
import contour;
import solids;
size(6cm,0);
currentprojection=orthographic(0.1,0.1,1);
real rc = 1, hc = 2, c = rc/hc;
//这里的scale(real,real,real)在plain_prethree宏包里
draw(shift(hc*Z)*scale(rc,rc,-hc)*unitcone, lightcyan);
triple Os = (0.5,0.5,1);
real r = 0.5;
draw(shift(Os)*surface(sphere(r), 60), surfacepen=invisible, meshpen = yellow);

real g(pair z){return (sqrt(z.x^2+z.y^2))/c;}
real f(pair z){
real x=z.x;
real y=z.y;
return (x-Os.x)^2+(y-Os.y)^2+(g(z)-Os.z)^2-r^2;
}
//contour生成两曲面交线的XY平面投影(消去了z维度),然后使用lift把该投影提升到圆锥曲面上.
draw(lift(g,contour(f,(-1,-1),(1,1),new real[]{0})),linewidth(2bp)+red);
*/

/*
//曲面上画曲线，利用lift函数画两个曲面的交线
//这里是个简单的已知曲线方程的例子
import graph3;
size(200,0);
currentprojection=orthographic(4,0,2);
real R=3;
real a=1;
triple f(pair z){return ((a*cos(z.y)+R)*cos(z.x),(a*cos(z.y)+R)*sin(z.x),a*sin(z.y));}
draw(surface(f, (0,0), (2pi,2pi),40,20,Spline),lightgreen);

triple c(real t){return ((a*cos(8*t)+R)*cos(t),(a*cos(8*t)+R)*sin(t),a*sin(8*t));}
draw(graph(c,0,2pi,operator..),linewidth(2)+red);
*/

/*
//material的应用
import solids;
size(6cm,0);
currentprojection=orthographic(1,2,2);
surface s=surface(sphere(1,n=10));
write(s.s.length);
material m=material(diffusepen=grey,ambientpen=yellow,
```

```
emissivepen=black,specularpen=orange);
material[] p={m,invisible,orange};
p.cyclic=true;
draw(s,p);
*/

/*
//自动画经纬线，目前来说只能用于旋转体，也就是revolution的数据结构
import solids;
size(200);
currentprojection=perspective(45,45,20);
path p =(0,3)--(2,3)..(3,3.5)..(4,4.5)..(4.5,6)..(4,8)..(1,10)..(2,12);
path3 generator = path3(p, YZplane);
revolution R = revolution(generator);
draw(R.silhouette(64), red+1bp); //silhouette用于画轮廓图，理论上是二维的

skeleton s; //新建骨架类
//加入横向线
R.transverse(s, reltime(R.g,0.5), currentprojection); //返回长度是一半时的实数值
R.transverse(s,0.5*length(R.g),currentprojection); //返回段数是一半时的实数值
draw(s.transverse.front,blue);
draw(s.transverse.back,blue+dashed);

//加入纵向线
R.longitudinal(s, currentprojection);
draw(s.longitudinal.front,pink+1pt);
draw(s.longitudinal.back,pink+1pt+linetype("8 8",8));
*/

/*
//solids.asy提供了skeleton(骨架)的数据结构
struct skeleton
{
struct curve
{
path3[] front; //正对观察者的曲线路径
path3[] back; //背对观察者的曲线路径
}
curve transverse; //横向线
curve longitudinal; //纵向线
}
*/

/*
//自动画虚线
size(200);
import solids; //可以自动画虚线的宏包
currentprojection=orthographic(5,4,2);
revolution sphere=sphere(1);
draw(surface(sphere), green+opacity(0.2));
draw(sphere, m=7,blue);
*/

/*
//Asymptote提供了不少常见的旋转体函数,unitsphere, unitcone, unitcylinder, unitsolidcone,
```

```
unithemisphere,unitfrustum
import three;
import graph3;
size(500);
currentprojection = orthographic(5,4,3);
draw(unitdisk,lightgreen+opacity(1));

xaxis3("x", Bounds());
yaxis3("y", Bounds());
zaxis3("z", Bounds());
*/

/*
//二维函数确定的直线，然后作旋转体
size(200,0);
import solids; //revolution函数在solid宏包里
currentprojection = perspective(8,8,6);
real f(real x){return x^2;}
path p = graph(f, 0, 1.5, 20, operator..);
path3 p3 = YZ*path3(p);
revolution pe = revolution(p3);
draw(surface(pe), lightyellow, black);
*/

/*
//旋转体
size(200);
import solids; //revolution函数在solid宏包里
currentprojection=perspective(5,4,4);
//Circle(center, radius, normal, vga)
draw(shift(3X)*Circle(O,1,Y,32),red+2);
//revolution(path3, rotate_axis, start_angle, end_angle);
revolution torus = revolution(shift(3X)*Circle(O,1,Y,32), Z, 90, 360);
draw(surface(torus), lightgreen+opacity(0.8));
*/

/*
//旋转体
size(200);
import solids;
currentprojection=orthographic(3,3,1);
path g= (0, 0.65){right}
..{up}(0.45, 1)
--(0.5, 1){down}
..{left}(0.1,0.6)
{right}..{down}(0.15,0.55)
..{down}(0.075,0.35){down}
..{down}(0.075, 0.2)
..(0.15,0.15){down}
..{left}(0.1,0.1){right}
..{right}(0.4,0.05)
--(0.4,0){left}
..{left}(0,0.05);
pen[] colors={red,green,blue};
colors.cyclic=true;
```

```
for(int i=0;i {
path3 p=path3(subpath(g,i,i+1),YZplane);
revolution R=revolution(p); //revolution公转，传入path3，返回revolution
draw(surface(R),colors[i]+1bp);
}
*/

/*
//旋转体基底，在xy平面上画的path
size(200);
import graph;
path g= (0, 0.65){right}
..{up}(0.45, 1)
--(0.5, 1){down}
..{left}(0.1,0.6)
{right}..{down}(0.15,0.55)
..{down}(0.075,0.35){down}
..{down}(0.075, 0.2)
..(0.15,0.15){down}
..{left}(0.1,0.1){right}
..{right}(0.4,0.05)
--(0.4,0){left}
..{left}(0,0.05);
pen[] colors={red,green,blue};
colors.cyclic=true;
for(int i=0;i draw(subpath(g,i,i+1),colors[i]+1bp); //一小段一小段的画，分别给颜色
}
xaxis("$x$",Arrow());
yaxis("$y$",Arrow());
*/

/*
//Steiner’s Roman Surface
size(200);
import graph3;
currentlight=White;
triple f(pair z){
triple t=expi(z.x,z.y);
return (t.y*t.z,t.z*t.x,t.x*t.y);
}
draw(surface(f,(0,0),(pi,2pi),50,50,Spline),meshpen=currentpen,palegrey);
*/

/*
//使用球坐标绘图 球体
import graph3;
import palette;
size(200);
triple f(pair z){return expi(z.x,z.y);}
surface s=surface(f,(0,0),(pi,2pi),30,Spline);
//主要还是填充颜色函数palette的使用,abs代表根据长度大小填充颜色
s.colors(palette(s.map(abs), Rainbow()));
draw(s, lightgreen+opacity(0.8), meshpen=nullpen, nolight);
*/
```

```
/*流程图
size(200);
real margin = 2mm;
pair A = (0,2), B = (-2,0), C = (2,0), D =(-2,-2), F = (2,-2);
object one = draw("1", box, A, margin); //object的创建很重要
object four = draw("4", roundbox, B, margin);
object five = draw("5", ellipse, C, margin);
object two = draw("2", box, D, margin);
object three = draw("3", roundbox, F, margin);
*/

/*
//球坐标系和直角坐标系的转换函数:
colatitude(triple v); //返回三维向量v与z轴的夹角(角度),也就是theta
longitude(triple v,bool warn=true); //返回v和XY平面投影与x的夹角,也就是phi

real polar(triple v); //返回v与z轴正方向夹角(弧度)
azimuth(triple v); //返回v在XY平面投影与x轴的夹角(弧度)

triple dir(real colatitude, real longitude); //输入theta,phi返回单位向量
real length(triple v); //返回v的长度
*/

/*
//球坐标系
import graph3;
size(200);
currentprojection=perspective(8,2,4);
pen bg=gray+opacity(0.2);
draw(surface((2,0,0)--(2,0,2)--(0,0,2)--(0,0,0)--cycle),bg,bg);
draw(surface((0,2,0)--(0,2,2)--(0,0,2)--(0,0,0)--cycle),bg,bg);
draw(surface((2,0,0)--(2,2,0)--(0,2,0)--(0,0,0)--cycle),bg,bg);

real rho=1.5;
real theta=30;
real phi=60;
//这里的arc已经扩展到三维情况了，使用的就是球坐标系
draw(arc(O,rho,0,phi,90,phi),dashed);
draw(O--dir(90,phi)*rho, dashed);
draw("ρ", O--rho*dir(theta,phi), Arrow3);
dot(rho*dir(theta,phi),red+2);
draw("ϕ" ,arc(O, rho/3, 90, 0, 90, phi), Arrow3(HookHead2));
draw("θ", arc(O, rho/3, 0,0,theta,phi), N+0.3E, Arrow3(HookHead2));
arrow("(ρ,θ,ϕ)", rho*dir(theta,phi), length = 20, E);
*/

/*
//中文立体文字 //贴在表面上
settings.tex="xelatex";
texpreamble("\usepackage{xeCJK}"); //使用xeCJK中文包
texpreamble("\setCJKmainfont{SimSun}"); //设置字体

import graph3 ;
size(300);
currentprojection=orthographic(-20,-24,292);
```

```
real f(pair z){return z.x^2-z.y^2;}
surface surf=surface(f,(-6,-4),(6,4),30,30);
//画出表面
draw(surf,palegray,nolight,meshpen=paleblue+1pt);
string ctex="哈哈哈哈中文\TeX";
real uoffset=0.3*30;
real voffset=0.3*30;
draw(surf.uequals(uoffset),red+dashed+1.2pt); //画出v等于uoffset时的先
draw(surf.vequals(voffset),green+dashed+1.2pt);
//画出贴在表面上的中文，height代往外突出的高度
draw(surface(scale(0.2)*ctex,surf,uoffset,voffset,height=0.2),yellow);
*/

/*
//双曲柱面 利用的还是extrude函数
import contour; //等高线宏包
import graph3;
currentprojection=orthographic(5,10,10);
currentlight=(10,10,2);
size(200);
real a = 4;
real b = 3;
real f(real x, real y)
{
return x^2/a^2-y^2/b^2;
}
guide[][] g = contour(f,(-10,-10),(10,10),new real[]{1}, 50); //50是分辨率
for(path p:g[0])
{
draw(extrude(p,4Z),palered);
draw(extrude(p,-4Z),palered);
draw(path3(p),red+2pt);
}
axes3("𝒙","𝒚","𝒛",Arrow3());
*/

/*
//柱状图形以及立体文字
//extrude函数可以画出柱状曲面
import three;
size(200);
currentprojection=orthographic(-2,-2,1);
path[] g = texpath("𝑳𝒂𝑻𝒆𝑿");
for(path p:g)
{
draw(path3(p), lightblue+1pt+opacity(0.5)); //底线
draw(extrude(p, 2Z), lightcyan); //表面
draw(shift(2Z)*path3(p),lightblue+1pt+opacity(0.5)); //顶线
}
*/

/*
//使得背景可以如同曲面那样着色.
size(200);
```

```
import solids;
currentprojection=orthographic(5,4,3);
revolution Sp=sphere(1);
draw(Sp,m=5,blue+1pt);
draw(Sp.silhouette(100),blue+1pt);
currentpicture.add
(
new void(frame f, transform3 t, picture pic, projection P)
{
draw(f,surface(invert(box(min(f,P),max(f,P)),min3(f),P),new pen[] {orange,green,yellow,red}));
}
);
*/

/*
//修改opengl渲染得到的图形的背景颜色
size(200);
import three;
currentlight.background=blue+opacity(0.1);
viewportmargin=(10,10);
currentprojection=orthographic(5,4,3);
draw(unitsphere,yellow);
*/

/*
//palette，map,以及自定义颜色的综合例子
import graph3;
import palette;
size(200,200,keepAspect=true);
real w=0.4;
real f(triple t) {return sin(t.x);}
triple f1(pair t) {return (cos(t.x)-2cos(w*t.y),sin(t.x)-2sin(w*t.y),t.y);}
triple f2(pair t) {return (cos(t.x)+2cos(w*t.y),sin(t.x)+2sin(w*t.y),t.y);}
triple f3(pair t) {return (cos(t.x)+2sin(w*t.y),sin(t.x)-2cos(w*t.y),t.y);}
triple f4(pair t) {return (cos(t.x)-2sin(w*t.y),sin(t.x)+2cos(w*t.y),t.y);}
surface s1=surface(f1,(0,0),(2pi,10),8,8,Spline);
surface s2=surface(f2,(0,0),(2pi,10),8,8,Spline);
surface s3=surface(f3,(0,0),(2pi,10),8,8,Spline);
surface s4=surface(f4,(0,0),(2pi,10),8,8,Spline);
s1.colors(palette(s1.map(f),Rainbow()));
s2.colors(palette(s2.map(f),Rainbow()));
s3.colors(palette(s3.map(f),Rainbow()));
s4.colors(palette(s4.map(f),Rainbow()));
draw(s1);
draw(s2);
draw(s3);
draw(s4);
*/

/*
//自定义着色函数方案
import graph3;
import palette; //着色库
size(200, IgnoreAspect);
```

```
currentprojection=orthographic(1,0.8,1);
real f(pair z)
{
real u = z.x;
real v = z.y;
return (u/2+v)/(2+cos(u/2)*sin(v));
}
surface s = surface(f, (2,2), (14,14), 40, 20, Spline);
//map接受一个输入为triple输出为实数值函数，
//s.map(zpart)返回real[][],第一个维度代表第几个patch,第二个纬度代表函数处理triple后的实数结果
//Rainbow()返回pen[]
//palette返回pen[][]
//函数的主要意思是按z轴填充颜色
s.colors(palette(s.map(zpart), Rainbow()));
draw(s);
*/

/*
//着色例子,对于生成的函数曲线
import graph3;
currentprojection=orthographic(1,-2,1);
currentlight=(1,-1,0.5);
size(200,0);
real sinc(pair z)
{
real r=2pi*abs(z);
return r != 0 ? sin(r)/r : 1;
}
surface sf=surface(sinc,(-2,-2),(2,2),Spline);
pen[] p= {cyan,magenta,yellow,green};
for(int i=0;i {
draw(surface(sf.s[i].external(),sf.s[i].internal(),p),nolight);
}
*/

/*
//对立方体surface进行截取路径染色
import three;
size(200);
surface s = unitcube;
pen[] ps = {red+opacity(0.5),yellow+opacity(0.5),black+opacity(0.5),cyan+opacity(0.5)};
for(int i = 0; i < s.s.length; ++i)
{
draw(surface(s.s[i].external(), s.s[i].internal(), ps));
}
*/

/*
//对球进行颜色处理
import graph3;
size(200);
currentprojection=orthographic(4,4,4);
surface sf=unitsphere;
for(int i = 0; i {
```

```
//可以把patch里面的external()理解为首尾点返回类型是path3 ,internal()理解为控制点
draw(surface(sf.s[i].external(), sf.s[i].internal(),new pen[]{cyan,magenta,yellow,green}), nolight);
}
dot(sf.s[0].external());
dot(sf.s[0].internal(), red);
*/

/*
//颜色处理,还是要对path3进行处理
import three;
size(200);
currentprojection=perspective(4,5,5);
real a=1,b=1,c=1;
path3[] g={
(0,0,0)--(a,0,0)--(a,b,0)--(0,b,0)--cycle,
(0,0,0)--(a,0,0)--(a,0,c)--(0,0,c)--cycle,
(a,0,0)--(a,b,0)--(a,b,c)--(a,0,c)--cycle,
(a,b,0)--(0,b,0)--(0,b,c)--(a,b,c)--cycle,
(0,b,0)--(0,0,0)--(0,0,c)--(0,b,c)--cycle,
(0,0,c)--(a,0,c)--(a,b,c)--(0,b,c)--cycle
};
pen[] p = {red+opacity(0.7), green+opacity(0.7), blue+opacity(0.7), black+opacity(0.7)};
for(path3 pa:g)
{
draw(surface(pa, p), nolight);
}
*/

/*
//总结下，着色处理可以在patch函数中和在surface函数中
//着色处理 patch接收参数triple[][]和pen[]
import three;
currentprojection=orthographic(1,1,5);
size(200);
triple[][] P=
{
{(-1,-1,-0.5), (-1,-0.3,0), (-1,0.3,0), (-1,1,-0.5)},
{(-0.3,-1,0), (-0.3,-0.3,0.5), (-0.3,0.3,0.5), (-0.3,1,0)},
{(0.3,-1,0), (0.3,-0.3,0.5), (0.3,0.3,0.5), (0.3,1,0)},
{(1,-1,-0.5), (1,-0.3,0), (1,0.3,0), (1,1,-0.5)}
};
surface s = surface(patch(P, new pen[] {red,green,blue,black}));
draw(s);
*/

/*
//着色处理和改变背景颜色,surface接受path3和pen数组参数
import three;
pen[] aaa = {red, green, black, white};
aaa.cyclic = true;
draw(surface(path3(polygon(5)), new pen[] {red, green, blue, black,white}));
draw(surface(path3(unitcircle), aaa));
*/

/*
```

```
//光照下的立体投影
import graph3;
currentprojection=orthographic(2,2,1);
size(200);
real R=2;
real a=1;
triple f(pair t) {
return ((R+a*cos(t.y))*cos(t.x),(R+a*cos(t.y))*sin(t.x),a*sin(t.y));
}
surface s=shift(3Z)*surface(f,(radians(90),0),(radians(345),2pi),8,8,Spline);
draw(s,lightgrey);
path3 pl=path3((-5,-5)--(5,-5)--(5,5)--(-5,5)--cycle);
draw(shift((-1e-3)*Z)*surface(pl),palered+opacity(0.5));
draw( planeproject(pl, -currentlight.position[0]) * s); //取光源的位置点
axes3("x","y","z", Arrow3);
*/

/*
//画三维物体的三视图投影
import graph3;
size(200);
currentprojection=orthographic(10,8,10);
real f(pair z) {
real u=z.x, v=z.y;
return (u/2+v)/(2+cos(u/2)*sin(v));
}
surface s=surface(f,(2,2),(14,14),Spline);
draw(s,palegrey);
//XY、YZ、ZX本身就是一个transform3类型，作用于path3
draw(surface(planeproject(XY*unitsquare3) * s ), lightgreen, nolight);
draw(surface(planeproject(YZ*unitsquare3) * s ), lightyellow, nolight);
draw(surface(planeproject(ZX*unitsquare3) * s ), lightblue, nolight);
axes3("x","y","z",Arrow3());
*/

/*
//three.asy宏包中预先定义了一些投影变换XY,YZ,ZX, ZY,YX,XZ
import graph3;
size3(200);
currentprojection = orthographic(4,6,3);
currentlight = nolight;
limits((0,0,0),(1.2,1.2,1.2));
draw(unitsquare3, linewidth(2pt));
draw(surface(XY*unitsquare3), red+opacity(0.5)); //这种变换直接换平面了
draw(surface(YZ*unitsquare3), green+opacity(0.5));
draw(surface(ZX*unitsquare3), blue+opacity(0.5));
axes3("x","y", "z", Arrow3);
*/

/*
//特定平面投影planeproject
import three;
size(8cm,0);
currentprojection=obliqueX;
```

```
currentlight=(0,2,1);
path3 plane = plane(12X,12Y,(-3,-3,0));
triple dir = (1,-1,4);
path3 p = (5,3,4)..(5,4,8)..(1,4,4)..(4,-2,3)..cycle;
draw(surface(plane),lightyellow+opacity(.5),blue); //blue是边框颜色
draw(surface(p),cyan+opacity(.9),red+1pt,nolight);
transform3 proj = planeproject(plane, dir);
draw(proj*p, dir, blue);
draw(O--dir, Arrow3);
for(int i =0;i {
draw(point(p,i)--point(proj*p,i), dotted, Arrow3(4));
}
*/

/*
//3D变换 transform3
shift(triple v);
xscale3(real x);
yscale3(real y);
zscale3(real z);
scale3(real s);
scale3(real x, real y, real z);
rotate(real angle, triple v); //绕着以原点为起点的方向v旋转angle角度
rotate(real angle, triple u, triple v); //绕着轴u--v旋转angle角度
reflect(triple u, triple v, triple w); //关于uvw确定的平面进行平面反射

//沿着dir方向，往过o点且法向量为n的平面作投影。
planeproject(triple n,triple o=o, triple dir = n);
//给定一个封闭路径p确定的平面，沿着dir方向投影。
planeproject(path3, triple dir=normal(p));
*/

/*
size(300);
import three;
//currentprojection=orthographic(20,-10,8);
currentprojection=perspective(20,-10,8);
triple a_orth = 2X;
triple b = 2Y;
triple a_par = 0.5 * b;
triple a = a_orth + a_par;
draw((4,4,0)--(-4,4,0)--(-4,-4,0)--(4,-4,0)--cycle);
draw(surface(O--0.3Y--(0.3Y+0.3Z)--0.3*Z--cycle),lightyellow,nolight);
draw(surface(O--(-0.3)*X--(-0.3X+0.3Z)--0.3Z--cycle),lightyellow,nolight);
draw(surface(O--(-0.3Y)--(-0.3Y-0.3X)--(-0.3X)--cycle),lightyellow,nolight);
draw(O--a_orth,Arrow3(DefaultHead2(normal=Z))); //箭头法向量为Z
draw(O--b,red,Arrow3(DefaultHead2(normal=Z)));
draw(O--a,red,Arrow3(DefaultHead2(normal=Z)));
draw(O--cross(a,b),red,Arrow3(DefaultHead2()));
draw(a_orth--a_orth+a_par,Arrow3(DefaultHead2(normal=Z)));
draw(O--(-0.3)*b);
draw(O--(-0.3)*a_orth);
draw(a--a+3*a_par,deepgreen+dashed,Arrow3(DefaultHead2(normal=Z)));
draw(a_orth--a_orth+(-3)*a_par,deepgreen+dashed);
```

```
int[] I={-2,-1,0,1,2,3};
I=reverse(I);
for(int i: I)
{
if(i!=0 && i!=1)
//Label中的string可以转到YZ平面。
draw(Label(YZ()*format("x_{%d}",find(I==i)+1),EndPoint,align=S,red),
O--a_orth+i*a_par, deepgreen, Arrow3(DefaultHead2(normal=Z)));
}
*/

/*
//grid3使用综合例子
import graph3;
import grid3;
import contour; //等值线库,不加也问题不大
import palette; //颜料库,不加也问题不大
size(8cm, 7cm, IgnoreAspect);
currentprojection = orthographic(-10,-10,12);
limits((0,0,0),(5,10,12));
real f(pair z){return (z.x+z.y)/(2+cos(z.x)*sin(z.y));}
surface s = surface(f,(0,0),(5,10),50,Spline);
draw(s,green);

grid3(new grid3routines [] {XYXgrid, ZXgrid(10), ZYgrid(5)},
Step=2, //大线间隔
step=1, //小线间隔
pGrid=new pen[] {red, blue, black}, //红蓝黑分别对应三个平面的大坐标线
pgrid=new pen[] {0.5red, lightgray, lightgray});

xaxis3(Label("𝒙",position=MidPoint,align=SE),Bounds(Min,Min),OutTicks());
yaxis3(Label("𝒚",position=MidPoint,align=SW),Bounds(Min,Min),OutTicks(Step=2));
//把下面的z坐标轴定位于x=0,y=10.并且标签朝向为Y(0,1,0)方向。
zaxis3(Label("𝒛",position=EndPoint,align=N+W),XYEquals(0,10),InTicks(beginlabel=false,endlabel=false,
Label(align=Y)));
*/

/*
//grid3宏包提供了三维栅格的功能
import grid3;
size(200,0,IgnoreAspect);
currentprojection=orthographic(0.25,1,0.25);
limits((-2,-2,0),(0,2,2));
//grid3函数带有默认参数，通常情况下grid3(XYZgrid)就可以
grid3(XYZgrid);
//grid3(pic=currentpicture,gridroutine=XYZgrid(pos=Relative(0)),N=0,n=0,Step=0,step=0,begin=true,end=true,pGrid=grey,pgric
xaxis3(Label("𝒙",position=EndPoint,align=S), Bounds(Min,Min), OutTicks());
yaxis3(Label("𝒚",position=EndPoint,align=S), Bounds(Min,Min), OutTicks());
zaxis3(Label("𝒛",position=EndPoint,align=(0,0.5)+W), Bounds(Min,Min), OutTicks(beginlabel=false));
*/

/*
//画出球体的包围框
import three;
import graph3;
```

```
size(200);
draw(unitsphere, lightcyan);
xaxis3("$x$", Bounds(), lightred+dashed);
yaxis3("$y$", Bounds());
zaxis3("$z$", Bounds());
*/

/*
//Bounds坐标轴包围框 graph3宏包提供
import graph3;
import contour;
size(7.5cm,0);
size3(7.5cm,IgnoreAspect);
real f(pair z)
{return 2z.x^2-z.x+z.y^2;}
currentprojection=orthographic(-10,-10,200);
limits((0,0,0),(10,10,300));
xaxis3(Label("$x$", position=MidPoint, align=SE), OutTicks(Step=2));
yaxis3(Label("$y$", position=MidPoint, align=SW), OutTicks(Step=2));
//Bounds表示画出坐标轴包围框，接收Min,Max,Both分别对应平行该轴的四条线。
zaxis3(Label("$z=2x^2-x+y^2$", position=EndPoint, align=3N+E),
Bounds(Min,Max),InTicks(Step=100, Label(align=Y)));
draw(surface(f,(0,0),(10,10),nx=5,Spline),lightgray,meshpen=black+thick(),nolight);
*/

/*画立方体
import palette;
import three;
import graph3;
size(500);
draw(unitcube, lightblue+opacity(0.7), meshpen = thick()+black, nolight);
axes3("x","y","z", (0,0,0),(1.5,1.5,1.5) ,Arrow3);
path3 l = (0,0,0)--(1,0,0);
path3 m = (1,0,0)--(1,1,0);
path3 n = (1,1,0)--(1,1,1);
draw(Label("$a=30$", position = MidPoint),shift(0,1.01,0)*l, Bars3(15,(0,0,1)), align = (0,5,0), p = red);
draw(Label("$b=30$", position = MidPoint),shift(0.01,0,0)*m, Bars3(15,(0,0,1)), align = (5,0,0),p = red);
draw(Label("$c=30$", position = MidPoint),shift(0.01,0.01,0)*n, Bars3(15,(1,1,0)), align = (5,0,0),p = red);
*/

/*
//带tick的3D坐标轴
import graph3;
size(8cm,0);
currentprojection=orthographic(1,1,1);
limits( (0,-2,0),(2,2,2) );
xaxis3("$x$", InTicks());
yaxis3("$y$", InTicks(Label, 10, 3)); //轴分为10等份，每等份再细分3等份。
zaxis3("$z$", OutTicks(beginlabel=false), p = red, arrow = Arrow3);//取消坐标重叠
*/

/*
//3d坐标轴
import graph3;
size(200,IgnoreAspect);
```

```
currentprojection = orthographic(1,1,1);
axes3("$x$", "$y$", "$z$",(0,-2,0),(2,2,2),Arrow3);
*/

/*
//紧凑参数方程
import graph3;
size(200,0);
real R=2;
real a=1;
triple f(pair t)
{
return ((R+a*cos(t.y))*cos(t.x),(R+a*cos(t.y))*sin(t.x),a*sin(t.y));
}
pen p = rgb(0.2,0.5,0.7);
draw(surface(f,(0,0),(2pi,2pi),30,15,Spline), lightgray, meshpen=p+thick()); //thick让网格更清晰,nullpen不画表面
*/

/*
//参数方程
import graph3;
size(200);
currentprojection = orthographic(-4,-4,5);
real R = 12;
triple f(pair t)
{
real u=t.x;
real v=t.y;
real x=(R+u*cos(v/2))*cos(v);
real y=(R+u*cos(v/2))*sin(v);
real z=u*sin(v/2);
return (x,y,z);
}
material m = material(diffusepen=grey, ambientpen=yellow, emissivepen=black,specularpen=orange); //设置材料
draw(surface(f, (-pi,0),(pi, 2pi), 50, Spline), m);
*/

/*
import graph3;
size(0,200);
currentprojection = orthographic(3,2,4);
currentlight = (5,3,0);
real f(pair z) {return z.y^2-z.x^2;}
draw(surface(f, (-3,-2),(3,2), nx=32, Spline), lightblue+opacity(0.3), meshpen=blue); //样条插值数nx*nx个三角形,spline光滑插值过度
*/

/*
size(200);
import graph;
pair A = (-3,-2), B = (3,-2), C =(3,2), D=(-3,2);
fill(A--B--C--D--cycle,lightgray);
label("$(-3,-2)$",A,SW);
label("$(3,2)$",C,NE);
```

```
xtick("$-3$",-3); //坐标轴具体位置贴上标签
xtick("$3$",3);
ytick("$-2$",-2);
ytick("$2$",2);
dot(A,red);
dot(C,red);
xaxis("$x$",Arrow,above=true); //在图片最上方
yaxis("$y$",Arrow,above=true);
*/

/*
//画除某个曲面的切平面
import three;
import graph3;
size(200,0);
triple[][] P =
{
{(-1,-1,-0.5), (-1,-0.3,0), (-1,0.3,0), (-1,1,-0.5)},
{(-0.3,-1,0), (-0.3,-0.3,0.5), (-0.3,0.3,0.5), (-0.3,1,0)},
{(0.3,-1,0), (0.3,-0.3,0.5), (0.3,0.3,0.5), (0.3,1,0)},
{(1,-1,-0.5), (1,-0.3,0), (1,0.3,0), (1,1,-0.5)}
};
surface sf = surface(patch(P));
draw(sf, palecyan+opacity(0.8));
draw(sequence(new path3(int i){return sf.s[i].external();}, sf.s.length), orange+1bp);
real t = 0.5;
path3 u = sf.s[0].uequals(t);
path3 v = sf.s[0].vequals(t);
draw(u,red);
draw(v,red);

//画切平面
triple po = point(u,t); //两条路径的交点
dot(po, blue);
triple dU = dir(u,t); //取路径为1/2时的方向向量
triple dV = dir(v,t);
draw(surface(plane(3dU,3dV,po+(-1.5dU)+(-1.5dV))),yellow+opacity(0.5));
triple dX=0.5dU+dV;
triple dY=0.5dU+0.5dV;
draw(po--po+dX,Arrow3(DefaultHead2(normal=Z)));
draw(po--po+dY,Arrow3(DefaultHead2(normal=Z)));
draw(po--po+dX+dY,Arrow3(DefaultHead2(normal=Z)));
axes3("$x$","$y$","$z$", Arrow3);
*/

/*
//从Bezier补丁返回我们需要的一些曲面上的元素
import three;
currentprojection = orthographic(2, -4, 5);
size(200);
triple[][] P=
{
{(-1,-1,-0.5), (-1,-0.3,0), (-1,0.3,0), (-1,1,-0.5)},
{(-0.3,-1,0), (-0.3,-0.3,0.5), (-0.3,0.3,0.5), (-0.3,1,0)},
```

```
{(0.3,-1,0), (0.3,-0.3,0.5), (0.3,0.3,0.5), (0.3,1,0)},
{(1,-1,-0.5), (1,-0.3,0), (1,0.3,0), (1,1,-0.5)}
};

surface sf = surface(patch(P)); //把上面四条路径打包成一个补丁
draw(sf, cyan+opacity(0.8));
draw(sf.s[0].uequals(0.5), red); //只有一条补丁
draw(sf.s[0].vequals(0.5), red); //只有一条补丁
draw(sequence(new path3(int i){return sf.s[i].external();}, sf.s.length), orange+1bp);
*/

/*
//surface函数，输入为一条封闭路径，输出为Bezier曲面
import three;
size(200);
path3 p = ( (1,2,2)..(3,9,2)..(4,4,4)..(2,3,0)..cycle );
surface sf = surface(p, new pen[]{cyan, yellow, magenta, black});
draw(sf, nolight); //把光照效果关闭 nolight
dot(sf.s[0].internal());
dot(sf.s[0].external());
*/

/*
import three;
size(200);
triple[][][] P=
{
{
{(-2.7,0,1.65),(-2.7,0.3,1.65),(-3,0.3,1.65),(-3,0,1.65)},
{(-2.7,0,1.875),(-2.7,0.3,1.875),(-3,0.3,2.1),(-3,0,2.1)},
{(-2.3,0,1.875),(-2.3,0.3,1.875),(-2.5,0.3,2.1),(-2.5,0,2.1)},
{(-1.6,0,1.875),(-1.6,0.3,1.875),(-1.5,0.3,2.1),(-1.5,0,2.1)}
},
{
{(-1.6,0,1.875),(-1.6,-0.3,1.875),(-1.5,-0.3,2.1),(-1.5,0,2.1)},
{(-2.3,0,1.875),(-2.3,-0.3,1.875),(-2.5,-0.3,2.1),(-2.5,0,2.1)},
{(-2.7,0,1.875),(-2.7,-0.3,1.875),(-3,-0.3,2.1),(-3,0,2.1)},
{(-2.7,0,1.65),(-2.7,-0.3,1.65),(-3,-0.3,1.65),(-3,0,1.65)}
},
{
{(-2.7,0,1.65),(-2.7,-0.3,1.65),(-3,-0.3,1.65),(-3,0,1.65)},
{(-2.7,0,1.425),(-2.7,-0.3,1.425),(-3,-0.3,1.2),(-3,0,1.2)},
{(-2.5,0,0.975),(-2.5,-0.3,0.975),(-2.65,-0.3,0.7275),(-2.65,0,0.7275)},
{(-2,0,0.75),(-2,-0.3,0.75),(-1.9,-0.3,0.45),(-1.9,0,0.45)}
},
{
{(-2,0,0.75),(-2,0.3,0.75),(-1.9,0.3,0.45),(-1.9,0,0.45)},
{(-2.5,0,0.975),(-2.5,0.3,0.975),(-2.65,0.3,0.7275),(-2.65,0,0.7275)},
{(-2.7,0,1.425),(-2.7,0.3,1.425),(-3,0.3,1.2),(-3,0,1.2)},
{(-2.7,0,1.65),(-2.7,0.3,1.65),(-3,0.3,1.65),(-3,0,1.65)}
}
};
pen[] colors={cyan,cyan,cyan,cyan};
for(int i=0; i < P.length; ++i)
```

```
{
draw(surface(patch(P[i])),colors[i]);
}
*/

/*
//Bezier曲面片
import three;
currentprojection = orthographic(2,-4,5);
size(200);
triple[][] P =
{
{(-1,-1,-0.5),(-1,-0.3,0),(-1,0.3,0),(-1,1,-0.5)},
{(-0.3,-1,0),(-0.3,-0.3,0.5),(-0.3,0.3,0.5),(-0.3,1,0)},
{(0.3,-1,0),(0.3,-0.3,0.5),(0.3,0.3,0.5),(0.3,1,0)},
{(1,-1,-0.5),(1,-0.3,0),(1,0.3,0),(1,1,-0.5)}
};
surface sf=surface(patch(P)); //patch函数把二元triple类型修补
draw(surface(sf), cyan+opacity(0.8));
draw(sequence(new path3(int i){return sf.s[i].external();}, sf.s.length), orange+1bp);
*/

/*
//三维参数曲线,可以通过graph函数把该参数方程专转成path3,
import graph3;
size(0,200);

real x(real t) {return cos(2pi*t);}
real y(real t) {return sin(2pi*t);}
real z(real t) {return t;}
path3 p = graph(x,y,z,0,2.7,operator..);

draw(p,Arrow3);
*/

/*
//三维变换 平移
import graph3;
size(200);
defaultpen(linewidth(2));
path p = (1,0)..(0,1)..(-1,0)..(0,-1)..cycle;
draw(shift(X+Y) * XY() * path3(p), red);
draw(shift(Y+Z) * YZ() * path3(p), green);
draw(shift(Z+X) * ZX() * path3(p), blue);
axes3(Arrow3(10));
*/

/*
//使用path3函数直接转换二维路径
import three;
size(200);
path p = (1,0)..(0,1)..(-1,0)..(0,-1)..cycle;
path3 pXY = path3(p, XYplane);
path3 pYZ = path3(p, YZplane);
path3 pZX = path3(p, ZXplane);
```

```
draw(pXY);
draw(pYZ);
draw(pZX);
dot(pXY,red);
dot(pYZ,red);
dot(pZX,red);
*/

/*
//3d曲线
import three;
size(200);
path3 g=(1,0,0)..(0,1,1)..(-1,0,0)..(0,-1,1)..cycle;
draw(g,green);
dot(g,red);
draw((-1,-1,0)--(1,-1,0)--(1,1,0)--(-1,1,0)--cycle, blue);
*/

/*
// 3d作图的数据类型
triple //三元实数组
path3 //三维路径
guide3 //待画出的三维路径
surface //曲面,与二维的graph对应
transform3 //三维变换,包括shift xscale3 yscale3 zscale3 scale3 rotate reflect
*/

/*
import three;
currentlight=light((10,10,10),(0,0,0)); //光源的位置
draw(unitcube, opacity(.5)+green);
*/

/*
//两相交平面自动消隐
size(7.5cm,0);
import three;
currentprojection=obliqueX;
currentlight=light((8,10,2),(8,0,8));
triple v1=(4,0,0),
v2=(0,6,0),
p0=(-2,-3,0);
path3 pl1=plane(v1,v2,p0);
path3 pl2=rotate(45,Y)*pl1;
draw(surface(pl1),yellow);
draw(surface(pl2),blue+opacity(.5));
draw(-v2/2--v2/2,2bp+red);
*/

/*
//斜投影
import three;
size(8cm,0);
currentprojection = obliqueX;
triple v1 = (4,0,0), v2 = (0,6,0), p0 = (-2,-3,0);
```

```
path3 pl = plane(v1,v2,p0); //根据点和向量画平面

draw(surface(pl), yellow);
draw(Label("$\vec{v_1}$"),p0--p0+v1,Arrow3);
draw(Label("$\vec{v_2}$"),p0--p0+v2,N,Arrow3);
dot("$p_0$",p0,NW);
*/

/*
//正交投影,平行投影，大小不变。能保持一个面的精确形状，无法很好表示立体特征
import graph3;
currentprojection = orthographic(1,1,1);
size(200);
limits((0,0,0), (1.2,1.2,1.2));
draw(unitcube,lightgreen);
axes3("$x$","$y$","$z$",Arrow3);
*/

/*
//透视投影,相当于有一台camera从固定点拍摄，和人的视角很接近
import graph3;
currentprojection = perspective(1,1,1); //设置初始化camera观察点
size(200);
limits((0,0,0), (1.2,1.2,1.2));
draw(unitcube, green);
axes3("$x$", "$y$", "$z$", Arrow3);
*/

/*
//3d有关的宏包 three graph3 solid tube grid3 contour3 obj labelpath3

//画简单的立方体
import three;
size(200);
defaultrender = render(compression=Zero, merge=true); //有损压缩参数为0,merge表示渲染前隐没节点提高显示速度
draw(unitcube, lightblue);
*/

/*
//画树的代码
size(200);
real branchrotation=60; //设置角度波动中值
real offset=180-branchrotation; //设置偏移角度
real thinning=0.7; //树枝大小缩放比例
real shortening=0.8; //树枝长度缩放比例
srand(seconds()); //根据时间生成随机数列
void tree(pair A,pair B,int n,real size)
{
pair C,D; //左右分支
real thickness=size; //树枝大小
real r_angle=offset+(branchrotation*(unitrand())); //120+[0~60];
C=B+shortening*(rotate(r_angle,B)A-B);
real l_angle=offset+(branchrotation(unitrand())); //120+[0~60];
D=B+shortening*(rotate(-l_angle,B)*A-B);
```

```
if(n>0)
{
draw(A--B,linewidth(thickness));
thickness=thinning*thickness; //改变树枝大小
tree(B,C,n-1,thickness);
tree(B,D,n-1,thickness);
}
else
{
draw(Label(scale(0.5)*"6", position = EndPoint),A--B,linewidth(thickness));
thickness=thinning*thickness;
draw(Label(scale(0.5)"6", position = EndPoint),B--C,linewidth(thickness));
draw(Label(scale(0.5)"6", position = EndPoint),B--D,linewidth(thickness));
}
}
tree((0,0),(0,1),10,2mm);
*/

/*
//分形图 其实就是一种递归图
size(300,0);
defaultpen(linewidth(1pt)+fontsize(20));
real shortening = 0.5;
void tree(pair A, pair B, int n)
{
pair C,D,M;
C = B + shortening*(rotate(90,B)A - B);
D = B + shortening(rotate(-90,B)A - B);
M = B + shortening(rotate(180,B)*A - B);
if(n>0)
{
draw(A--B);
tree(B,C,n-1);
tree(B,D,n-1);
tree(B,M,n-1);
}
else
{
draw(A--B);
draw(B--C);
draw(B--D);
draw(B--M);
}
}
tree((0,0), (0,1), 4);
tree((0,0), (1,0), 4);
tree((0,0), (-1,0), 4);
tree((0,0), (0,-1), 4);
*/

/*
//模拟徒手绘图的效果
import trembling; //手绘图的宏包
import geometry;
```

```
size(200);
pair[] compass={E,S,W,N,NE,NW,SE,SW};
for(pair z: compass)
{
path p=tremble(angle=8,frequency=0.1,random=100).deform(z..(0,0)); //手绘图效果
write(length(p));
draw(p,blue,MidArrow);
}
triangle T=triangleabc(2*sqrt(3),2*sqrt(3),2*sqrt(3));
draw(shift(-sqrt(3),-1)*T);
draw(unitcircle,red);
*/

/*
//随机曲线的生成
size(200);
pen colorpen=blue;
pair end=(0,0);
pair direction = E;
srand(seconds()); //按时间生成随机数序列

void randomwalk(int n)
{
if(n==0) return;
else
{
real angle = rand()%360;
real lambda = 1;
pair begin = end;
end = begin+scale(lambda)*direction;
path randompath = begin--end;
draw(randompath, colorpen);
direction = rotate(angle)*direction;
if(n==1000) dot(begin, red);
randomwalk(n-1);
}
}
randomwalk(1000);
*/

/*
//彩色路径文字
texpreamble("\usepackage{CJKutf8}
\AtBeginDocument{

\begin{CJK*}{UTF8}{gbsn}}
\AtEndDocument{\clearpage\end{CJK*}
}");

size(200);
path p = reverse(arc((0,0),1,20,160));
string[] fonts = {"中","华","人","民","共","和","国","万","岁"};
pen[] colors = {lightblue, lightcyan, lightgray, lightgreen,lightgrey,lightmagenta,lightolive,lightred,lightyellow};
colors.cyclic = true; //画笔循环
for(int i = 0; i < fonts.length;++i)
```

```
{
label(scale(2)*fonts[i], point(p, i/(fonts.length-1)*length(p)), colors[i]);//很好理解
}
*/

/*
//弯曲路径上的标签 中文标签
texpreamble("\usepackage{CJKutf8}
\AtBeginDocument{

\begin{CJK*}{UTF8}{gbsn}}
\AtEndDocument{\clearpage\end{CJK*}
}");

size(200);
import labelpath; //需要导入标签路径库
path p = reverse(rotate(-90)*yscale(2)*xscale(4)*unitcircle); //把路径翻转，主要是因为圆的路径从(1,0)开始逆时针走。
labelpath("中华人民共和国万岁", p, red);
*/

/*
settings.tex = "xelatex";
usepackage("xeCJK");
texpreamble("\setCJKmainfont{SimSun}");
*/

/*
//各种循环语句的应用
size(200);
int n = 12;
int k = 5;
defaultpen(linewidth(1)+fontsize(20));
pair P(int i){return dir(90+k*(360/n)*i);}
Label L(int i){return Label(format("A_{%d}", i+1), align=P(i));} //format格式化操作字符串
pair[] star = sequence(P,n);
Label[] starlabel = sequence(L,n);
dot(starlabel, star);
draw(operator--(...star)--cycle); //不是很理解这里的表达式
*/

/*
//读入数据绘图,第一种方法

import graph;
size(200, IgnoreAspect);
file f = input("C:\Users\Flew\Desktop\我的坚果云\test\data.txt");
write(f);
pair[] a;
while(true)
{
real x = f;
real y = f;
if(eof(f)) break;
a.push((x,y));
}
```

```
draw(graph(a), red);
dot(a);
xaxis(BottomTop, LeftTicks());
yaxis(LeftRight, RightTicks());

//读入数据绘图，第二种方法
import graph;
size(200, IgnoreAspect);
file filename = input("C:\Users\Flew\Desktop\我的坚果云\test\data.txt").line();
write(filename);
real[][] A=filename.dimension(0,0); //创建二维数据，表明filename的数据是二维的
A = transpose(A); //把A的二维数组转置
real[] x = A[0];
real[] y = A[1];
guide g = graph(x,y);
draw(g,red);
pair[] a = pairs(x,y); //这里要特别注意pairs函数的使用
dot(a);
*/

/*
//画线程图,流程图
size(200);
real margin = 2mm;
pair A = (0,2), B = (-2,0), C = (2,0), D =(-2,-2), F = (2,-2);
object one = draw("1", ellipse, A, margin); //object的创建很重要
object four = draw("4", ellipse, B, margin);
object five = draw("5", ellipse, C, margin);
object two = draw("2", ellipse, D, margin);
object three = draw("3", ellipse, F, margin);

add(new void(picture pic, transform t)
{
draw(pic,Label("two",align=W), point(two,SW,t){SW}..{SE}point(two,NW,t),Arrow); //关键是这个point提取了object
的点
draw(pic,Label("three",align=E), point(three,SE,t){SE}..{SW}point(three,NE,t),Arrow);
draw(pic,Label("four",align=W), point(four,NW,t){NW}..{NE}point(four,SW,t),Arrow);
draw(pic,Label("five",align=E), point(five,NE,t){NE}..{NW}point(five,SE,t),Arrow);
draw(pic,point(one,SW,t){SW}..{SW}point(four,NE,t),Arrow);
draw(pic,point(one,SE,t){SE}..{SE}point(five,NW,t),Arrow);
draw(pic,point(four,E,t){E}..{E}point(five,W,t),Arrow);
draw(pic,Label("1/2",align=N),point(two,NE,t){NE}..{SE}point(three,NW,t),Arrow);
draw(pic,Label("1/4",align=N),point(three,SW,t){SW}..{NW}point(two,SE,t),Arrow);
});
*/

/*
//分段函数
import graph;
size(0,100);

pair[] z = {(0,1/4), (1/4,1), (1,2+3/4), (2+3/4,4+1/4), (4+1/4,5)};
real[] Y = {1,0.92,0.8325,0.7217,0.5171};

for(int i = 0; i < Y.length; ++i)
```

```
{
draw((z[i].x, Y[i])--(z[i].y, Y[i]), red);
dot((z[i].x, Y[i]));
dot((z[i].y, Y[i]), UnFill); //圆点不填充,空心
}

axes("$x$", "$y$", (0,0), (5.5,1.4), Arrow(HookHead, 2));
*/

/*
import graph;
size(0,100);
real u(real x) {return abs(x-floor(x+1/2));}

typedef real func(real);

func f(int n)
{
return new real (real x) {return u(4^n*x)/4^n;};
}
int n = 20;
real Sum(real x)
{
real s = 0;
for (int i = 0; i < n;++i)
{
s =s +f(i)(x);
}
return s;
}
draw(graph(Sum,-1,1,operator--),red);
*/

/*
//数项级数表示的函数
import graph;
size(200, IgnoreAspect);

typedef real func(real); //定义一种函数类型

func f(int n)
{
return new real (real x){return (n*x - 1/2 - floor(n*x - 1/2));}; //使用了匿名函数
}

int n = 35;

real Sum(real x)
{
real s = 0;
for (int i = 0; i < n;++i)
{
s = s + f(i)(x);
}
return s;
}
```

```
draw(graph(Sum,0,2,operator--), red);
*/

/*
//graph宏包画极坐标图
size(200);
import graph;

real f(real r){return 2*r;}
draw(polargraph(f,0,2pi,operator..),red);

int n = 4;

for(int i = 1; i < n; ++i)
{
draw(circle((0,0), i*5), gray+linetype("5 5")); //linetype("n m")n为实体线长，m为空位置长
}
for(int i = 0;i<360;i+=30)
{
  draw(Label(string(i), EndPoint), (0,0)--dir(i)*3*5);

}
*/

/*
//graph宏包，画能量图正弦余弦
import graph;
unitsize(4cm,8cm);
defaultpen(1.5pt+fontsize(20pt)+Helvetica()); //Helvetica是一种字体
real f(real x){return sin(2pi*x)^2*exp(-x);}
real g(real x){return cos(2pi*x)^2*exp(-x);}

draw(graph(f,0,4,operator..),red,Label("ϕ_{f_1} vs ϕ_{f_11} "));
draw(graph(g,0,4,operator..),blue+linetype("2 2"), Label("ϕ_{f_6} vs ϕ_{f_11} "));
xaxis("Frequency(kHz)",BottomTop, LeftTicks()); //BottomTop上下都有坐标轴,LeftTicks打得小短杆朝向
yaxis("Cross Power(a.u.)",LeftRight, RightTicks());

xequals(1, blue+Dotted); //x等于1的地方画虚线
xequals(2, blue+Dotted);
xequals(3, blue+Dotted);
yequals(0.5, blue+Dotted);

add(legend(),point((0,0)), NE, UnFill); //point((0,0))是画布正中心的绝对点,legend()表示加图例
*/

/*
//graph宏包,画函数曲线图
size(200,IgnoreAspect); //忽略等比例缩放
import graph; //曲线宏包
import patterns; //画阴影部分的宏包

add("hatch",hatch(H=2mm,dir=NE,gray));
real f(real t) {return 1/2+atan(t-2)/pi;}
real a = -4;
real b = 2;
```

```
real c = 6;
path p = (-4,0)--(0,0)--(0,1)--(2,1)--(2,f(b))..(graph(f,b,a))--cycle; //画出阴影部分
fill(p,pattern("hatch"));

draw(graph(f,a,c,operator..), linewidth(1bp));
draw((0,1)--(c,1));
draw((b,0)--(b,1), dashed);
draw((0,f(b))--(b,f(b)), dashed);
label("$p = F(t)$", (0,f(b)), W);
label("$p = F^{-1}(p)$", (b,0), S);
label("$O$", (0,0), S);
label("$1$", (0,1), W);
axes(Label("$x$", align=2E), Label("$u$", align=2N),(-4,0),(6,1.1),Arrow(HookHead,2));//x轴从-4到6,,,Y轴从0到1.1
*/

/*
//画出矢量图，这个例子我还没有理解，不理解的点在下面都标注了。
import slopefield;
picture base;
size(base, 200,200);
real func(real x){return 2x;}
add(base, slopefield(func,(-3,-3),(3,3),20,Arrow)); //这个slopefield函数到底是干什么的
draw(base, curve((0,0), func, (-3,-3), (3,3)), red); //还有这个curve函数
draw(base, box((0,0),(1,1)),blue);
add(base.fit()); //fit函数对picture进行了什么的处理

clip(base, unitsquare);
add(scale(0.5)*base.fit(), (150,20));
draw((34,17)--(150,60),Arrow);
*/

/*
size(200);
pair z0 = (0,0);
pair z1 = (3,0);
pair c0 = (-1,2);
pair c1 = (4,3);

path p = z0..controls c0 and c1..z1;
draw(p,red);

dot("$Z_0(0,0)$", z0, S);
dot("$Z_1(3,0)$", z1, S);
dot("$C_0(-1,2)$", c0, N);
dot("$C_1(4,3)$", c1, N);

write(length(p));
write(precontrol(p,0));
write(postcontrol(p,0));
write(precontrol(p,1));
write(postcontrol(p,1));
*/

/*
precontrol(path p,int t);
postcontrol(path p, int t);
```

```
length(path p); //返回分段数目
point(path p, real t); //得到t时刻对应的点
*/

/*
//三次贝塞尔曲线
import graph;
size(500);
pair z0 = (0,0);
pair z1 = (20, 0);
pair c0 = (5, 10);
pair c1 = (15,10);
real t = 1/3;
pair a0 = interp(z0,c0,t);
pair a1 = interp(c0,c1,t);
pair a2 = interp(c1,z1,t);
pair b0 = interp(a0,a1,t);
pair b1 = interp(a1,a2,t);
pair z = interp(b0,b1,t);

draw(z0--c0--c1--z1);
draw(a0--a1--a2);
draw(b0--b1);

pair f(real t){return (1-t)^3*z0 + 3*t*(1-t)^2*c0 + 3*t^2*(1-t)*c1 + t^3*z1;}
draw(graph(f,0,1),red);

dot("$Z_0$", z0);
dot("$Z_1$", z1);
dot("$A_0$", a0);
dot("$A_1$", a1);
dot("$A_2$", a2);
dot("$B_0$", b0);
dot("$B_1$", b1);
dot("$Z$", z);
arrow("$t = \frac{1}{3}$", z, SE);
*/

/*贝塞尔曲线的原理，二次参数曲线
import graph;
size(200);
pair Z0=(0,0),Z1=(4,0),C=(1,3);
real t = 1/3;
pair A = interp(Z0, C, t);
pair B = interp(C, Z1, t);
pair Z = interp(A, B, t);
draw(Z0--C--Z1);
draw(A--B);
arrow("$t = \frac{1}{3}$", Z, SE);
pair f(real t){return (1-t)^2*Z0+2t*(1-t)*C+t^2*Z1;}
draw(graph(f, 0, 1), red);

dot("$C$", C, N);
dot("$A$", A, W);
dot("$Z_0$", Z0, S);
```

```
dot("$Z_1$", Z1, SE);
dot("$B$",B ,NE);
*/

/*与pair有关的一些函数
conj(pair z) 复共轭
length(pair z) 返回长度
abs(pair z) 返回长度
angle(pair z) 返回z的幅角，区间[-π，π]内取值
degrees(pair z,bool warn=true) 返回幅角,单位度。
unit(pair z) 返回单位向量
expi(real angle) 返回弧角度angle为方向的单位向量
dir(real degrees) 返回角度degrees为方向的单位向量
xpart(pair z) 返回z.x
ypart(pair z) 返回z.y
dot(pair z, pair w) 返回点积
minbound(pair z,pair w) 返回最小边界
maxbound(pair z,pair w) 返回最大边界
*/

/*
size(8cm,0);
path A=(0,0)--(2,1)--(2,3)--(0,2)--cycle;
path B=reflect((0,0),(0,1))*A;
path C=rotate(180,(0,0))*A;
path D=reflect((0,0),(1,0))*A;

picture pic;
filldraw(pic, A^^C, lightblue+white, black);
filldraw(pic, B^^D, lightred+white, black);

for(int i = 0; i < 9; ++i)
{
for (int j = 0; j < 6; ++j)
{
add(shift(i*4,j*6) * pic);
}
}
*/

/*自己实现的叠加立方体版本
size(500);
picture pic_cube;
pair O = (0,0);
pair B = (0,6);
pair A = (4, 3);
pair C = (-4,3);
path up_face = O--A--B--C--cycle;
path left_face = O--C--(C-(0,5))--(O-(0,5))--cycle;
path right_face = O--A--(A-(0,5))--(O-(0,5))--cycle;
filldraw(pic_cube ,up_face, lightgray,black);
filldraw(pic_cube, left_face, lightred, black);
filldraw(pic_cube, right_face, lightblue, black);

for (int j = 0; j < 10; ++j)
```

```
{
for(int i = 0; i < 10; ++i)
{
if(j%2==0)
{
add(shift(i*8,j*(11.0/2.0))pic_cube, above = false);
}
else
{
add(shift(4,3) shift(i*8,j*(11.0/2.0))*pic_cube);
}
```

```
  }
```

```
}
*/

/*
size(500);
path base_square = (0,0)--(10,0)--(10,10)--(0,10)--cycle;

for (int i = 0; i < 8; ++i)
{
for (int j = 0; j < 8; ++j)
{
draw(shift(i*10,j*10)*base_square);
}
}
*/

/*
size(500);
draw(unitcircle);
draw(xscale(.5) * unitcircle,green);
draw(scale(5)*unitcircle,red);
*/

/*
size(500);
int n = 8;
real step = 360/n;
path base_pic = ellipse((0,0), 5,3);

for(int i = 0; i < n; ++i)
{
draw(rotate(i*step)*base_pic, black);
}
*/

/*
defaultpen(magenta);
size(500);
int n = 5;
for(int i = 0; i < n; ++i)
{
```

```
dot(dir(i*360/n), red+6pt);
}
*/

/*对称轴反射 以a--b为对称轴
reflect(pair a, pair b);
*/

/*
//旋转变换
import geometry;
size(500);
pair P = (0,0);
pair A = (20,20);
path PA = P--A;
draw(Label("$A$",position=EndPoint),PA, Arrow);
draw(Label("$B$",position=EndPoint),rotate(60)*PA,Arrow);
draw(rotate(150)*PA);
draw(Label("$60^\circ$",blue),arc(A,P,rotate(60)*A, 5), red, Arrow);

dot("$P$", P, S);
*/

/*
//错切变换
size(500);
draw(unitsquare, black);
draw(slant(1.5)*unitsquare,red); //其中tan(倾斜角度)=参数因子
dot((0.5,0.5), lightgreen);
dot( slant(1.5)*(0.5,0.5), lightred);
*/

/*平移变换
size(500);
filldraw(unitsquare,palegreen);
filldraw(shift(1,0)*unitsquare,mediumgreen);
filldraw(shift(2,1)*unitsquare,heavygreen);
*/

/*
size(500);
pair O = (0,0);
path p1 = circle(0,50);
path p2 = circle(0,100);
path p3 = circle(0,150);
path p4 = circle(0,200);

fill(p1^^p2^^p3^^p4, evenodd+magenta); //最外面一层一定填充
*/

/*
//clip实战
size(500,500);
pair A = (-50,0);
pair B = (50,0);
path circleA = circle(A, 80);
```

```
path circleB = circle(B, 80);
fill(circleA, 0.8white);
fill(circleB, 0.8white);

picture intersect;
fill(intersect, circleA, 0.6white);
clip(intersect, circleB);
add(intersect);

draw(circleA, black);
draw(circleB, black);

draw(Label("$A \cap B$", position = BeginPoint), (0,-100)--(0,0),black ,Arrow);
label("$A$", A, UnFill(5mm));
label("$B$", B, UnFill(5mm));
*/

/*
//clip添加图层.
size(500);
draw(circle((0,0),50));
path tri = (-50,0)--(50,0)--(50,50)--cycle;
filldraw(tri, lightgray, black);

picture dashedline;
draw(dashedline, circle((0,0), 50), dashed+red);
clip(dashedline, tri);
add(dashedline);
*/

/*
//clip指令
size(500);
import patterns;
//draw(unitcircle, red);
add("name", hatch(50,NW,red));
filldraw(unitsquare, pattern("name"));

clip(currentpicture, unitcircle, false);
*/

/*
//子图，图层,above指令
size(500);
picture pic;
picture fig;
filldraw(pic, unitcircle, green);
filldraw(fig, unitsquare, red);
add(pic);
add(fig, above = false);
*/

/*
//画阴影,需要调用patterns库
hatch阴影
import patterns;
```

```
size(500);
add("name", hatch(50 ,NW,red));
fill(unitsquare, pattern("name"));
*/

/*
//隐函数表达需要调用contour库(等值线库)
//隐函数方程为x^3+y^3-3axy=0
import graph;
import contour;
size(500);
real a = 1;
real f(pair t)
{
real x = t.x;
real y = t.y;
return x^3+y^3-3a*x*y;
}

axes("𝒙", "𝒚", Arrow);
real c[] = {0,10,20}; //需要几条等值线
guide[][] g=contour(f,(-4,-4),(4,4),c,100);
draw(g, red);
*/

/*
//参数曲线
import graph;
size(500);
pair f(real t)
{
real x = cos(t);
real y = sin(t);
return (x, y);
}

xaxis("𝒙", Arrow);
yaxis("𝒚", Arrow);
draw(graph(f, -4, 4, operator--));
*/

/*
//画自定义函数的曲线图
import graph;
size(500);
real f1(real x){return x^3;};
real f2(real x){return x^2;};
real f3(real x){return x;};

guide g1 = graph(f1, 0, 2, operator..);
guide g2 = graph(f2, 0, 2, operator..);
guide g3 = graph(f3, 0, 2, operator..);

xaxis(Label("𝒙", position = EndPoint, align = N), Arrow);
yaxis("𝒚", Arrow);
```

```
draw(Label("$y=x^3$", EndPoint, E) ,g1, blue);
draw(Label("$y=x^2$", EndPoint, E), g2, red);
draw(Label("$y=x$", EndPoint, E), g3, green);
*/

/*
//自定义函数,需要graph曲线库,作曲线图
import graph;
size(500);
real f(real x){return x^2;};
path p = graph(f,-2,2);
draw(p,red);
xaxis("$x$", Arrow);
yaxis("$y$", Arrow);
*/

/*
//画直角标志
import geometry;
size(500);
pair A=(0,0),B=(250,0),C=(250,250),D=(0,250);
draw(A--B--C--D--cycle, black+10);

perpendicular(A,NE,A--B);
perpendicular(B,NE,B--C);
perpendicular(C,NE,C--D);
perpendicular(D,NE,D--A);
*/

/*//画角度弧线,需要geometry
import geometry; //导入后arc(pair,pair,pair,int)重载了
//正常版本arc(pair,real r, real stangle, real endangle);
size(500);
defaultpen(linewidth(2));
pair O=(20,20);
pair P=(200,0);
pair Q=(200,200);
draw(P--O--Q);
dot("$P$",P, black+5);
dot("$O$",O, SW,black+5);
dot("$Q$",Q,black+5);
draw(arc(P,O,Q,50),Arrow);
draw(arc(Q,O,P,80), red, Arrow);
*/

/*//画弧线
size(500);
defaultpen(linewidth(10));
pair O=(0,0);
draw(arc(O,200,0,120),red);
draw(arc(O,200,120,180),lightgreen);
*/

/*//正多边形
size(500);
```

```
draw(unitcircle);
filldraw(polygon(5), lightgreen, black);
*/

/*//内置形状
import graph;
size(500);
pair min=(-20,-20);
pair max=(20,20);
pair O=(0,0);
pair A=(2,0);
draw(box(min,max), red);
draw(ellipse(O,20,10), magenta);
draw(circle(O,10), green);
*/

/*//画贝赛尔曲线
size(500);
pair O = (0,0);
pair P = (20, 20);
draw(O--P, black+3,Arrow);
pair A=(0,5),B=(5,10),C=(10,5),D=(0,-10);
pair E=(-10,5), F=(-5, 10);
filldraw(A{up}..{right}B..{down}C..(6,-5)..D..(-6,-5)..{up}E..{right}F..{down}A..cycle, pink, red);
draw((-20,-20)--O, black+3);
*/

/*//画点
size(500);
draw(E..N..W..S..cycle);
dot("N", N, N);
dot("S", S, S);
dot("E", E, E);
dot("W", W, W);
*/

/*//分段函数
size(500);
pair A=(0,0), B=(4,0), C=(4,4), D=(0,4);
pair A_B = interp(A,B,1/4),
B_C = interp(B,C,1/4),
C_D = interp(C,D,1/4),
D_A = interp(D,A,1/4);

draw(A--B--C--D -- cycle, black+10);
draw(A_B -- B_C -- C_D -- D_A -- cycle, red+10);
*/

/*取中点函数
size(500);
pair A = (0, 0), B = (4, 0), C = (1, 2);
draw( A--B--C--cycle, black+10);
filldraw( midpoint(A--B) -- midpoint(B--C) -- midpoint(A--C) -- cycle, (red)+10);
*/

/*//标签
```

```
size(500);
draw(Label("aaa", EndPoint), unitsquare, (opacity(0.5)+red)+1mm , Arrow);
*/

/*
size(500);
pair o = (0,0);
draw(Label("N", EndPoint, red), o--N, red, Arrow);
draw(Label("S", EndPoint), o--S, Arrow);
draw(Label("W", EndPoint), o--W, Arrow);
draw(Label("E", EndPoint), o--E, Arrow);

draw(Label("NE", EndPoint), o--NE, Arrow);
draw(Label("NW", EndPoint), o--NW, Arrow);
draw(Label("SE", EndPoint), o--SE, Arrow);
draw(Label("SW", EndPoint), o--SW, Arrow);

draw(Label("NNE", EndPoint), o--NNE, Arrow);
draw(Label("ENE", EndPoint), o--ENE, Arrow);
draw(Label("SSE", EndPoint), o--SSE, Arrow);
draw(Label("ESE", EndPoint), o--ESE, Arrow);

draw(Label("NNW", EndPoint), o--NNW, Arrow);
draw(Label("WNW", EndPoint), o--WNW, Arrow);
draw(Label("SSW", EndPoint), o--SSW, Arrow);
draw(Label("WSW", EndPoint), o--WSW, Arrow);
*/
```