Sample FrAid Programs

Is used to generate:

```
///////////////////////Aggregation//////////////////////////////////////

aggregation();
```

Is used to generate:aggregation1 aggregation2

(go to top)

```
//////////////////////////////Cobweb///////////////////////////////

r=1;
controlVar(r);
f(x)=r*x*(1-x);
cobweb(f);
```

Is used to generate:cobweb1

(go to top)

```
//////////////////////////////Cobweb///////////////////////////////

p=1;
controlVar(p);
f(x)= if x < 0 then 0 else
        if (x >= 0) & (x < 1/2) then p*x else
        if (x >= 1/2) & (x < 1) then -p*x+p else 0;
cobweb(f);
```

Is used to generate:cobweb2

(go to top)

```
//////////////////////////////Cobweb///////////////////////////////

r=1;
controlVar(r);
f(x)=r*x*(1-x);
g(x)=f(f(x));
cobweb(g);
```

Is used to generate:cobweb3

""

(go to top)

```
///////////////////////Color 3D/////////////////////////////////////

f1(x,y)=sin(abs(x)-abs(y));
color3D(f1);

f2(x,y)=cos(abs(x)+abs(y))*(abs(x)+abs(y));
color3D(f2);

f3(x,y)=cos(abs(x)+abs(y));
color3D(f3);

f5(x,y)=abs(cos(x^2+y^2))^(1/8);
color3D(f5);

f6(x,y)=x^2+y^2;
color3D(f6);
```

Is used to generate:color3d1 color3d2 color3d3 color3d4 color3d5

(go to top)

```
///////////////////////////Filter inverse, fiter
reverse////////////////////////
samplingF = 2048;     //Hz
samplingTime = 1;     //seconds
filterLength = 81;    //points
shape(x) = if x < ( samplingF / 2 ) / 3 then 1 else 0; //say limit to a
third of the interval
fr(x)=sampleN( shape, 0, 1, samplingF/2+1 ); //freq. response
//plot(fr);

fk(x)=ifft1(fr);                             //filter kernel 1
//plot(fk);

fkshr(x)=shrotS(fk,filterLength/2);
fktr(x)=truncateS(fkshr,0,filterLength-1);
fkn(x)=fktr(x)/sumS(fktr);                      //normalized kernel

blackmanW(x,filterLength) = 0.42 - 0.5  * cos( 2 * Pi * x / filterLength )
+ 0.08 * cos( 4 * Pi * x / filterLength ); //Blackman

wfk(x)=blackmanW(x/stepS(fkn),filterLength)*fkn(x);      //the Blackman
windowed filter

fkInv(x)=inverseFilter(wfk);                 //inverse
//fkInv(x)=reverseFilter(wfk);               //reverse

plot(wfk,fkInv);
```

```
//check the resulting filters frequency response
efr(x)=firResp(wfk);                      //estimated freq. response
efrInv(x)=firResp(fkInv);                      //estimated freq. response
length 2

plot(efr,efrInv,-.1,1.1,Pi+.1,-.5);
```

Is used to generate:freq_2fltr

(go to top)

```
///////////////////Compare window functions////////////////////////////////
samplingF = 2048;     //Hz
samplingTime = 1;     //seconds
filterLength = 41;    //points
shape(x) = if x < ( samplingF / 2 ) / 3 then 1 else 0; //say limit to a
third of the interval
fr(x)=sampleN( shape, 0, 1, samplingF/2+1 ); //freq. response
//plot(fr);
fk(x)=ifft1(fr);                             //filter kernel
//plot(fk);

fktr(x)=truncateS(shrotS(fk,filterLength/2),0,filterLength-1);
fkn(x)=fktr(x)/sumS(fktr);                      //normalized kernel
//plot(fkn);

hammingW(x) = 0.54 - 0.46 * cos( 2 * Pi * x / filterLength ); //Hamming
blackmanW(x) = 0.42 - 0.5  * cos( 2 * Pi * x / filterLength ) + 0.08 * cos(
4 * Pi * x / filterLength ); //Blackman

wfkh(x)=hammingW(x/stepS(fkn))*fkn(x);      //the Hamming windowed filter
wfkb(x)=blackmanW(x/stepS(fkn))*fkn(x);     //the Blackman windowed filter

//check the resulting filters frequency response
efr(x)=firResp(fkn);                        //estimated freq. response
non-windowed
efrh(x)=firResp(wfkh);                      //estimated freq. response
Hamming
efrb(x)=firResp(wfkb);                      //estimated freq. response
Blackman

plot(efr,efrh,efrb,-.1,1.1,Pi+.1,-.5);

db(x)=8.685890*log(x);

efrdb(x)=db(efr(x));                            //same as above but in db
efrdbH(x)=db(efrh(x));
efrdbB(x)=db(efrb(x));
plot(efrdb,efrdbH,efrdbB,-.1,1,Pi+.1,-70);

//alternatively do the same through the fourier transform of the filter
```

```
kernel

fknp(x)=padS(fkn,2^nextpow2(samplingF*samplingTime));  //pad to the proper
length
fkn1f(x)=fft1(fknp);                            //take fft
fkn1a(x)=abs(fkn1f(x))*samplingF/2;                 //take abs and
normalize
fkn1db(x)=db(fkn1a(x));                             //calculate in db

wfkp1(x)=padS(wfkh,2^nextpow2(samplingF*samplingTime)); //same for the
Hamming windowed kernel...
wfkh1f(x)=fft1(wfkp1);
wfkh1a(x)=abs(wfkh1f(x))*samplingF/2;
wfkh1db(x)=db(wfkh1a(x));

wfkbp(x)=padS(wfkb,2^nextpow2(samplingF*samplingTime)); //same for the
Blackman windowed kernel...
wfkb1f(x)=fft1(wfkbp);
wfkb1a(x)=abs(wfkb1f(x))*samplingF/2;
wfkb1db(x)=db(wfkb1a(x));

//plot(fkn1,wfkh1,wfkb1);
plot(fkn1a,wfkh1a,wfkb1a);
plot(fkn1db,wfkh1db,wfkb1db);
```

Is used to generate:freq 3fir freq 3fir db

(go to top)

```
//////////////////////////////Band-pass/Band-reject out of a High-Pass and
a Low-Pass//////////
samplingF = 2048;     //Hz
samplingTime = 1;     //seconds
filterLength = 81;    //points
shape(x) = if x < ( samplingF / 2 ) / 3 then 1 else 0; //say limit to a
third of the interval
fr(x)=sampleN( shape, 0, 1, samplingF/2+1 ); //freq. response
//plot(fr);

fk(x)=ifft1(fr);                             //filter kernel 1
//plot(fk);

fkshr(x)=shrotS(fk,filterLength/2);
fktr(x)=truncateS(fkshr,0,filterLength-1);
fkn(x)=fktr(x)/sumS(fktr);                      //normalized kernel

blackmanW(x,filterLength) = 0.42 - 0.5  * cos( 2 * Pi * x / filterLength )
+ 0.08 * cos( 4 * Pi * x / filterLength ); //Blackman
db(x)=8.685890*log(x);

wfk(x)=blackmanW(x/stepS(fkn),filterLength)*fkn(x);      //the Blackman
windowed filter
```

```
fkInv(x)=reverseFilter(wfk);             //make high pass out of the low
pass

plot(wfk,fkInv);

//check the resulting filters frequency response
efr(x)=firResp(wfk);                      //estimated freq. response
efrInv(x)=firResp(fkInv);
plot(efr,efrInv,-.1,1.1,Pi+.1,-.5);

//rf(x)=conv(wfk,fkInv);     //band-pass -- THIS DOESN'T WORK, LOTS OF
NOISE !!!
rf(x)=wfk(x)+fkInv(x);           //result filter band-reject
rfn(x)=rf(x)/sumS(rf);
plot(rfn);

//rffr(x)=firResp(rfn);
//plot(rffr,-.1,1.1,Pi+.1,-.5);

//check the response but doing fft on the kernel
fknp(x)=padS(rf,2^nextpow2(samplingF*samplingTime));  //pad to the proper
length
fkn1f(x)=fft1(fknp);                                //take fft
fkn1a(x)=abs(fkn1f(x))*samplingF/2;                     //take abs and
normalize
fkn1db(x)=db(fkn1a(x));                                 //calculate in db
plot(fkn1a);
plot(fkn1db);


//Since the conv line above doesn't work we can get bandpass by inversion
of bandreject
rfInv(x)=inverseFilter(rfn);
plot(rfInv);
rfInvFr(x)=firResp(rfInv);
plot(rfInvFr,-.1,1.1,Pi+.1,-.5);
```

Is used to generate:freq br db

(go to top)

```
//////////////Spectrogram, oscillogram, histogram, Fourier transform in
real time///////////
sf=10000;
f(x)=recordSound(sf);
plot( histogramS( f, 19 ) );
a=spectrum(f,512);
plotOption(a,"threshold",.1);
plot({abs(fft1(f))});
plot(f,0,.01,1,-.01);
```

Is used to generate:freq spectrum

(go to top)

```
/////////Time And Frequency////////
f(t,om)=E^(i*t*om);
plot3f(f);
```

Is used to generate:freq vs time1

(go to top)

```
///////////////////////Julia////////////////////////////////////

Re = -.766227;
Im =  .09699;

controlVar(Im,Re);
f(z)=z^2+Re+Im*i;
julia(f);
```

Is used to generate:julia1 julia2 julia3 julia4 julia5 julia6

(go to top)

```
/////////////////////////////////Mandelbrot///////////////////////////
clear();

a=2;
controlVar(a);

f(z,c)=z^a+c;

mandelbrot("zMandelbrotPlugInDemo2",f);
```

Is used to generate:mandelbrot1 mandelbrot10 mandelbrot2 mandelbrot3 mandelbrot4 mandelbrot5 mandelbrot6 mandelbrot7 mandelbrot8 mandelbrot9

(go to top)

```
///////////////////////Lorenz
2D////////////////////////////////////////////

lor1( x1, x2, x3, t ) = 10 * (x2 - x1);
lor2( x1, x2, x3, t ) = 28*x1 - x2 - x1*x3;
lor3( x1, x2, x3, t ) = x1*x2 - 8/3*x3;

rk( lor1,//the system
    lor2,
    lor3,
    0, 1, 0, //the initial condition
    0, //the start point
    100, //the end point
    10000, /*number of samples*/
    "_rk");

plot2(_rk_0,_rk_2,0,30);
```

Is used to generate:math_lorenz1

(go to top)

```
///////////////////////////////Plot2////////////////////////////////

f(x)=sin(2*x)+sin(3*x);
plot2(sin,f,0, 2*Pi);
```

Is used to generate:math plot2 1 math plot2 2

(go to top)

```
/////////////////////////////////Plot/////////////////////////////

plot(sin, cos, tan, atan );
```

Is used to generate:math plot 1

(go to top)

```
/////////////////////////////Rosseler 2D////////////////////////////////
clear();
a=.2; b=.2; c=5.7;
x0=0.001; y0=1; z0=0.001;
startP=0; endP=100; numberSamples=1000;

controlVar( a, b, c, x0, y0, z0, startP, endP );

rsslr22( x1, x2, x3, t, a1 ) = x1 + a1*x2;
rsslr33( x1, x2, x3, t, b1, c1 ) = b1 + x3*(x1-c1);

rsslr1( x1, x2, x3, t ) = -x2 - x3;
rsslr2( x1, x2, x3, t ) = rsslr22( x1, x2, x3, t, a );
rsslr3( x1, x2, x3, t ) = rsslr33( x1, x2, x3, t, b, c );

rk1(
rsslr1,//the system
rsslr2,
rsslr3,
x0, y0, z0, //the initial condition
startP, //the start point
endP, //the end point
numberSamples, /*number of samples*/
"_rk1");
```

```
plot2(_rk1_0,_rk1_1,0,100);
```

Is used to generate:math ross1

(go to top)

```
/////////////////////////////Newton////////////////////////////////////

f(x)=x^3-1;
df(x)=diff(f,0,x);
g(x)=x-f(x)/df(x);
julia(g);

re = 1;
im = 1;
controlVar( re, im );
f( x ) = x^3 - (re + im * i);
mydiff(x)=(f(x+.0000001)-f(x))/.0000001;
g(x)=x-f(x)/mydiff(x);
julia(g);


f( x, c ) = x^3 - c;
mydiff(x,c)=(f(x+.0000001,c)-f(x,c))/.0000001;
g(x,c)=x-f(x,c)/mydiff(x,c);
mandelbrot(g);
```

Is used to generate:newton1 newton2

(go to top)

```
///////////////////////////Orbit////////////////////////////

f(x,r)=r*x*(1-x);
orbit(f);

f(x,p)= if x < 0 then 0 else
        if (x >= 0) & (x < 1/2) then p*x else
        if (x >= 1/2) & (x < 1) then -p*x+p else 0;
orbit(f);
```

Is used to generate:one d orbit1 one d orbit10 one d orbit11 one d orbit2 one d orbit3 one d orbit4 one d orbit5 one d orbit6 one d orbit7 one d orbit8 one d orbit9

(go to top)

```
//////////////Dust clouds////////////////////
clear;
```

""

```
a = .2;
b = .99;
c = 1;

controlVar( a, b, c );

//F(x) = a*x + ( 1 - a )*( 2*x^2 / ( 1 + x^2 ));
//F(x) = a*x + c*sin(x);
//F(x) = a*x + c*cos(x);
//F(x) = a   + c*sin(x);
//F(x) = a*x + c*x^2/( 1 + abs(x) );

F(x) = if x >  1 then a*x + c*( x - 1 ) else
           if x < -1 then a*x + c*( x + 1 ) else
                          a*x;
fx(x,y)=b*y + F(x);
fy(x,y) = -x + F(fx(x,y));

plot  ( F );
plot3f( fx );
plot3f( fy );
orbit2( fx, fy );
```

Is used to generate:orbit_dust1 orbit_dust10 orbit_dust11 orbit_dust12 orbit_dust13 orbit_dust14 orbit_dust15 orbit_dust16 orbit_dust17 orbit_dust18 orbit_dust19 orbit_dust2 orbit_dust20 orbit_dust3 orbit_dust4 orbit_dust5 orbit_dust6 orbit_dust7 orbit_dust8 orbit_dust9

(go to top)

```
//////////////Gingerbreadman////////////////////
clear;

fx( x, y ) = 1 - y + abs( x );
fy( x, y ) = x;

orbit2( fx, fy );
```

Is used to generate:orbit_gngr_brd_man1 orbit_gngr_brd_man2 orbit_gngr_brd_man3

(go to top)

```
//////////Henon///////////////////////
clear;

a = 1.4;
b = .3;
controlVar( a, b );

fx( x, y ) = 1 - y - a * x^2;
```

```
fy( x, y ) = b * x;

orbit2( fx, fy );
```

Is used to generate:orbit_henon1

(go to top)

```
//////////Hopalong///////////////////////
clear;

a = -55;
b = -1;
c = -42;
controlVar( a, b, c );

fx( x, y ) = y - sign(x)*sqrt( abs ( b*x - c ));
fy( x, y ) = a - x;

orbit2( fx, fy );
```

Is used to generate:orbit_hopalong1 orbit_hopalong2 orbit_hopalong3 orbit_hopalong4 orbit_hopalong5

(go to top)

```
//////////Hopalong Next//////////////////////
clear;

a = -55;
b = -1;
c = -42;
controlVar( a, b, c );

fx( x, y ) = y - sign(x)*abs( sin(x)*cos(b)+c - x*sin(a+b+c) );
fy( x, y ) = a - x;

orbit2( fx, fy );
```

Is used to generate:orbit_hopalong_next1 orbit_hopalong_next2 orbit_hopalong_next3 orbit_hopalong_next4 orbit_hopalong_next5

(go to top)

```
//////////Martin///////////////////////
clear;

a = -Pi;
controlVar( a );
```

```
fx( x, y ) = y - sin( x );
fy( x, y ) = a - x;

orbit2( fx, fy );
```

Is used to generate:orbit_m1 orbit_m2 orbit_m3 orbit_m4 orbit_m5

(go to top)

```
////////Popcorn////////////////////////
clear;

h = .05;
controlVar( h );

fx( x, y ) = x - h * sin( y + tan( 3 * y ));
fy( x, y ) = y - h * sin( x + tan( 3 * x ));

orbit2( fx, fy );
```

Is used to generate:orbit_popcorn1 orbit_popcorn2

(go to top)

```
///////////////////////////////Lorenz///////////////////////////////////////
clear();
 lor1( x1, x2, x3, t ) = 10 * (x2 - x1);
 lor2( x1, x2, x3, t ) = 28*x1 - x2 - x1*x3;
 lor3( x1, x2, x3, t ) = x1*x2 - 8/3*x3;

 rk( lor1,//the system
 lor2,
 lor3,
 0, 1, 0, //the initial condition
 0, //the start point
 100, //the end point
 10000, /*number of samples*/
 "_rk");

 plot3(_rk_0,_rk_1,_rk_2,0,30);
```

Is used to generate:three_d_lorenz1

(go to top)

```
////////////////////////////Math 3D
curves//////////////////////////////////////
clear;

f1(x,y)=sin(abs(x)-abs(y));
plot3f(f1);
```

```
f2(x,y)=cos(abs(x)+abs(y))*(abs(x)+abs(y));
plot3f(f2);

f3(x,y)=cos(abs(x)+abs(y));
plot3f(f3);

f4(x,y)=-1/cos(x^2+y^2);
plot3f(f4);

f5(x,y)=abs(cos(x^2+y^2))^(1/8);
plot3f(f5);

f6(x,y)=x^2+y^2;
plot3f(f6);
```

Is used to generate:three d parabolic1 three d parabolic2 three d parabolic2a three d parabolic3 three d parabolic4 three d parabolic5 three d parabolic6 three d parabolic7 three d parabolic8

(go to top)

```
/////////////////////Rosseler//////////////////////////////////////////////
clear();
a=.2; b=.2; c=5.7;
x0=0.001; y0=1; z0=0.001;
startP=0; endP=100; numberSamples=1000;

controlVar( a, b, c, x0, y0, z0, startP, endP );

 rsslr22( x1, x2, x3, t, a1 ) = x1 + a1*x2;
 rsslr33( x1, x2, x3, t, b1, c1 ) = b1 + x3*(x1-c1);

 rsslr1( x1, x2, x3, t ) = -x2 - x3;
 rsslr2( x1, x2, x3, t ) = rsslr22( x1, x2, x3, t, a );
 rsslr3( x1, x2, x3, t ) = rsslr33( x1, x2, x3, t, b, c );

 rk1(
 rsslr1,//the system
 rsslr2,
 rsslr3,
 x0, y0, z0, //the initial condition
 startP, //the start point
 endP, //the end point
 numberSamples, /*number of samples*/
 "_rk1");

plot3("zPlot3PlugInDemo1",_rk1_0,_rk1_1,_rk1_2,0,100);
```

Is used to generate:three d ross1

(go to top)

```
///////////////////////////////Serpinski 3D////////////////////////////////
clear();
ff(r,a,b,x,y,z)=r*(  cos(b)*x + sin(a)*sin(b)*y + cos(a)*sin(b)*z );
gg(r,a,b,x,y,z)=r*(             cos(a)       *y - sin(a)       *z );
hh(r,a,b,x,y,z)=r*( -sin(b)*x + sin(a)*cos(b)*y + cos(a)*cos(b)*z );

r1=.5;
a1=0;
b1=0;
xOff1=0.2;
yOff1=0.4;
zOff1=0;
controlVar(r1, a1, b1, xOff1, yOff1, zOff1 );

f1(x,y,z)=ff(r1,a1,b1,x,y,z);
g1(x,y,z)=gg(r1,a1,b1,x,y,z);
h1(x,y,z)=hh(r1,a1,b1,x,y,z);

r2=.5;
a2=0;
b2=0;
xOff2=.55;
yOff2=0;
zOff2=0;
controlVar(r2, a2, b2, xOff2, yOff2, zOff2 );

f2(x,y,z)=ff(r2,a2,b2,x,y,z);
g2(x,y,z)=gg(r2,a2,b2,x,y,z);
h2(x,y,z)=hh(r2,a2,b2,x,y,z);

r3=.5;
a3=0;
b3=0;
xOff3=0.25;
yOff3=.15;
zOff3=.5;
controlVar(r3, a3, b3, xOff3, yOff3, zOff3 );

f3(x,y,z)=ff(r3,a3,b3,x,y,z);
g3(x,y,z)=gg(r3,a3,b3,x,y,z);
h3(x,y,z)=hh(r3,a3,b3,x,y,z);

r4=.5;
a4=0;
b4=0;
xOff4=0;
yOff4=0;
zOff4=0;
controlVar(r4, a4, b4, xOff4, yOff4, zOff4 );

f4(x,y,z)=ff(r4,a4,b4,x,y,z);
g4(x,y,z)=gg(r4,a4,b4,x,y,z);
h4(x,y,z)=hh(r4,a4,b4,x,y,z);
```

```
transform3(
"zTransform3PlugInDemo2",
f1,xOff1,
g1,yOff1,
h1,zOff1,
f2,xOff2,
g2,yOff2,
h2,zOff2,
f3,xOff3,
g3,yOff3,
h3,zOff3,
f4,xOff4,
g4,yOff4,
h4,zOff4,
"pyramid.3d");
```

Is used to generate:three d serpinski1 three d serpinski2

(go to top)

```
/////////////Tree 3D///////////////////////////////////////////////////////
clear;

ff(r,a,b,x,y,z)=r*(  cos(b)*x + sin(a)*sin(b)*y + cos(a)*sin(b)*z );
gg(r,a,b,x,y,z)=r*(             cos(a)       *y - sin(a)       *z );
hh(r,a,b,x,y,z)=r*( -sin(b)*x + sin(a)*cos(b)*y + cos(a)*cos(b)*z );

r1=.65; //the parameters I want to control for the first transform
a1=-.45;
b1=.69;
xOff1=0.9;
yOff1=0.7;
zOff1=0;
controlVar(r1, a1, b1, xOff1, yOff1, zOff1 ); //variable controller

f1(x,y,z)=ff(r1,a1,b1,x,y,z); //the first transform
g1(x,y,z)=gg(r1,a1,b1,x,y,z);
h1(x,y,z)=hh(r1,a1,b1,x,y,z);

r2=.5; //the parameters I want to control for the first transform
a2=-.2;
b2=-.66;
xOff2=-.9;
yOff2=0;
zOff2=0.75;
controlVar(r2, a2, b2, xOff2, yOff2, zOff2 ); //variable controller

f2(x,y,z)=ff(r2,a2,b2,x,y,z); //the first transform
g2(x,y,z)=gg(r2,a2,b2,x,y,z);
h2(x,y,z)=hh(r2,a2,b2,x,y,z);

r3=.8; //the parameters I want to control for the first transform
a3=0;
```

```
b3=.14;
xOff3=0.1;
yOff3=-.15;
zOff3=1.5;
controlVar(r3, a3, b3, xOff3, yOff3, zOff3 ); //variable controller

f3(x,y,z)=ff(r3,a3,b3,x,y,z); //the first transform
g3(x,y,z)=gg(r3,a3,b3,x,y,z);
h3(x,y,z)=hh(r3,a3,b3,x,y,z);

r4=.5; //the parameters I want to control for the first transform
a4=.9;
b4=.14;
xOff4=0;
yOff4=-1.2;
zOff4=0;
controlVar(r4, a4, b4, xOff4, yOff4, zOff4 ); //variable controller

f4(x,y,z)=ff(r4,a4,b4,x,y,z); //the first transform
g4(x,y,z)=gg(r4,a4,b4,x,y,z);
h4(x,y,z)=hh(r4,a4,b4,x,y,z);

transform3( f1,xOff1, //first
            g1,yOff1,
            h1,zOff1,
            f2,xOff2, //first
            g2,yOff2,
            h2,zOff2,
            f3,xOff3, //first
            g3,yOff3,
            h3,zOff3,
            f4,xOff4, //first
            g4,yOff4,
            h4,zOff4,
            "trunk.3d");
```

Is used to generate:three_d_tree1

(go to top)

```
//////////////////////////////////////Another Tree////////////////////////
clear();
ff(r,a,b,x,y,z)=r*(  cos(b)*x + sin(a)*sin(b)*y + cos(a)*sin(b)*z );
gg(r,a,b,x,y,z)=r*(             cos(a)       *y - sin(a)       *z );
hh(r,a,b,x,y,z)=r*( -sin(b)*x + sin(a)*cos(b)*y + cos(a)*cos(b)*z );

r1=.65;
a1=-.45;
b1=.69;
xOff1=0;
yOff1=0;
zOff1=0.4;
```

```
f1(x,y,z)=ff(r1,a1,b1,x,y,z);
g1(x,y,z)=gg(r1,a1,b1,x,y,z);
h1(x,y,z)=hh(r1,a1,b1,x,y,z);

r2=.5;
a2=-.2;
b2=-.66;
xOff2=0;
yOff2=0;
zOff2=0.75;

f2(x,y,z)=ff(r2,a2,b2,x,y,z);
g2(x,y,z)=gg(r2,a2,b2,x,y,z);
h2(x,y,z)=hh(r2,a2,b2,x,y,z);

r3=.8;
a3=0;
b3=0;
xOff3=0;
yOff3=0;
zOff3=1;

f3(x,y,z)=ff(r3,a3,b3,x,y,z);
g3(x,y,z)=gg(r3,a3,b3,x,y,z);
h3(x,y,z)=hh(r3,a3,b3,x,y,z);

r4=.5;
a4=.9;
b4=.14;
xOff4=0;
yOff4=0;
zOff4=0.6;

controlVar(
r1, a1, b1, xOff1, yOff1, zOff1,
r2, a2, b2, xOff2, yOff2, zOff2);
controlVar(
r3, a3, b3, xOff3, yOff3, zOff3,
r4, a4, b4, xOff4, yOff4, zOff4 );

f4(x,y,z)=ff(r4,a4,b4,x,y,z);
g4(x,y,z)=gg(r4,a4,b4,x,y,z);
h4(x,y,z)=hh(r4,a4,b4,x,y,z);

transform3(
"zTransform3PlugInDemo1",
f1,xOff1,
g1,yOff1,
h1,zOff1,
f2,xOff2,
g2,yOff2,
h2,zOff2,
f3,xOff3,
g3,yOff3,
h3,zOff3,
f4,xOff4,
g4,yOff4,
h4,zOff4,
"segment.3d");
```

Is used to generate:three_d_tree2

(go to top)

```
//////////////////Cantor/////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.3; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0.85;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.3; //the parameters I want to control for the first transform
a2=0;
xOff2=0.9;
yOff2=0.85;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2); //variable controller

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "line_horizontal.2d" );
```

Is used to generate:two_d_cantor1

(go to top)

```
/////////////////Hilbert easy  ///////////////////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.33; //the parameters I want to control for the first transform
a1=1.58;
xOff1=0.45;
yOff1=0;
```

```
f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.33; //the parameters I want to control for the first transform
a2=3.14;
xOff2=0.88;
yOff2=0.43;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.33; //the parameters I want to control for the first transform
a3=1.58;
xOff3=0.45;
yOff3=-0.44;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.33; //the parameters I want to control for the first transform
a4=6.28;
xOff4=0.45;
yOff4=-0.45;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

r5=.33; //the parameters I want to control for the first transform
a5=1.57;
xOff5=0.89;
yOff5=0;

f5(x,y)=ff(r5,a5,0,x,y); //the first transform
g5(x,y)=gg(r5,a5,0,x,y);

r6=.33; //the parameters I want to control for the first transform
a6=1.57;
xOff6=0.89;
yOff6=-0.445;

f6(x,y)=ff(r6,a6,0,x,y); //the second transform
g6(x,y)=gg(r6,a6,0,x,y);

r7=.33; //the parameters I want to control for the first transform
a7=0;
xOff7=0;
yOff7=0;

f7(x,y)=ff(r7,a7,0,x,y); //the first transform
g7(x,y)=gg(r7,a7,0,x,y);

r8=.33; //the parameters I want to control for the first transform
a8=3.14;
xOff8=0.9;
```

```
yOff8=0;

f8(x,y)=ff(r8,a8,0,x,y); //the second transform
g8(x,y)=gg(r8,a8,0,x,y);

r9=.33; //the parameters I want to control for the first transform
a9=0;
xOff9=0.9;
yOff9=0;

f9(x,y)=ff(r9,a9,0,x,y); //the second transform
g9(x,y)=gg(r9,a9,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3);
controlVar(r4,a4,xOff4,yOff4,r5,a5,xOff5,yOff5,r6,a6,xOff6,yOff6);
controlVar(r7,a7,xOff7,yOff7,r8,a8,xOff8,yOff8,r9,a9,xOff9,yOff9);

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,
           g4,yOff4,
           f5,xOff5,
           g5,yOff5,
           f6,xOff6,
           g6,yOff6,
           f7,xOff7,
           g7,yOff7,
           f8,xOff8,
           g8,yOff8,
           f9,xOff9,
           g9,yOff9,
           "line_horizontal.2d");

/////////////////Hilbert modified////use with the one before///////
//2d_hilbert2

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,
           g4,yOff4,
           f5,xOff5,
           g5,yOff5,
           f6,xOff6,
           g6,yOff6,
           f7,xOff7,
           g7,yOff7,
           f9,xOff9,
           g9,yOff9,
           "line_horizontal.2d");
```

Is used to generate:two d hilbert1 two d hilbert2

(go to top)

```
////////////////Ice//////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.5; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.35; //the parameters I want to control for the first transform
a2=-1.57;
xOff2=0.7;
yOff2=0.5;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.5; //the parameters I want to control for the first transform
a3=0;
xOff3=0.65;
yOff3=0;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.35; //the parameters I want to control for the first transform
a4=-4.7;
xOff4=0.7;
yOff4=0;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,
r4,a4,xOff4,yOff4);

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           "line_horizontal.2d");
```

Is used to generate:two d ice1

(go to top)

```
////////////////Koch//////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.3; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.3; //the parameters I want to control for the first transform
a2=0.8;
xOff2=0.4;
yOff2=0;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.3; //the parameters I want to control for the first transform
a3=-.85;
xOff3=0.7;
yOff3=0.3;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.3; //the parameters I want to control for the first transform
a4=0;
xOff4=0.9;
yOff4=0;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,r4,a4,xOff4,yOff4);

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           "line_horizontal.2d");
```

Is used to generate:[two_d_koch1]

[(go to top)]

```
/////////////////////Levi/////////////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.7; //the parameters I want to control for the first transform
a1=.785;
xOff1=0;
yOff1=0;
controlVar(r1, a1,xOff1, yOff1 ); //variable controller

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.7; //the parameters I want to control for the first transform
a2=-.78;
xOff2=0.65;
yOff2=0.65;
controlVar(r2,a2,xOff2,yOff2); //variable controller

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "line_horizontal.2d" );
```

Is used to generate:[two_d_levi1]

[(go to top)]

```
/////////Mandelbrot tree////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.5; //the parameters I want to control for the first transform
a1=0;
```

```
xOff1=-0.7;
yOff1=0.8;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.5; //the parameters I want to control for the first transform
a2=3.14;
xOff2=-0.7;
yOff2=0.75;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.5; //the parameters I want to control for the first transform
a3=0;
xOff3=0.75;
yOff3=0.8;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.5; //the parameters I want to control for the first transform
a4=3.14;
xOff4=0.75;
yOff4=0.75;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,r4,a4,xOff4,yOff4);

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           "t.2d");
```

Is used to generate:two_d_mandelbrot_tr1

(go to top)

```
////////mess//////////////
clear();

ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off;
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;
```

```
r1=.75;
a1=1.234;
xOff1=0;
yOff1=0.35;
controlVar(r1, a1,xOff1, yOff1 );

f1(x,y)=ff(r1,a1,0,x,y);
g1(x,y)=gg(r1,a1,0,x,y);

r2=.6;
a2=0;
xOff2=0.2;
yOff2=0.5;
controlVar(r2,a2,xOff2,yOff2);

f2(x,y)=ff(r2,a2,0,x,y);
g2(x,y)=gg(r2,a2,0,x,y);
iterFract(
"zIterFractPlugInDemo1",
f1,xOff1,
g1,yOff1,
f2,xOff2,
g2,yOff2,
"pentagon.2d" );
```

Is used to generate:two d mess1 two d mess10 two d mess11 two d mess2 two d mess3 two d mess4 two d mess5 two d mess6 two d mess7 two d mess8 two d mess9

(go to top)

```
////////mess multicontrol///////////////
clear();

ff(rx,ry,a,b,off,x,y)=rx*cos(a)*x - ry*sin(b)*y + off;
gg(rx,ry,a,b,off,x,y)=rx*sin(a)*x + ry*cos(b)*y + off;

rx1=.75;
ry1=.75;
a1=1.234;
b1=1.234;
xOff1=0;
yOff1=0.35;
controlVar(rx1, ry1,a1,b1,xOff1, yOff1 );

f1(x,y)=ff(rx1,ry1,a1,b1,0,x,y);
g1(x,y)=gg(rx1,ry1,a1,b1,0,x,y);

rx2=.6;
ry2=.6;
a2=0;
b2=0;
xOff2=0.2;
```

```
yOff2=0.5;
controlVar(rx2,ry2,a2,b2,xOff2,yOff2);

f2(x,y)=ff(rx2,ry2,a2,b2,0,x,y);
g2(x,y)=gg(rx2,ry2,a2,b2,0,x,y);

iterFract(
"zIterFractPlugInDemo1",
f1,xOff1,
g1,yOff1,
f2,xOff2,
g2,yOff2,
"pentagon.2d" );
```

Is used to generate: two d mess multi1 two d mess multi10 two d mess multi11 two d mess multi2 two d mess multi3 two d mess multi4 two d mess multi5 two d mess multi6 two d mess multi7 two d mess multi8 two d mess multi9

(go to top)

```
///////////Minkowski//////////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.25; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.25; //the parameters I want to control for the first transform
a2=-1.6;
xOff2=.33;
yOff2=.33;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.25; //the parameters I want to control for the first transform
a3=0;
xOff3=0.33;
yOff3=0.33;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.5; //the parameters I want to control for the first transform
a4=-1.6;
xOff4=0.66;
```

```
yOff4=0.33;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

r5=.25; //the parameters I want to control for the first transform
a5=0;
xOff5=0.66;
yOff5=-0.33;

f5(x,y)=ff(r5,a5,0,x,y); //the first transform
g5(x,y)=gg(r5,a5,0,x,y);

r6=.25; //the parameters I want to control for the first transform
a6=-1.6;
xOff6=1;
yOff6=0;

f6(x,y)=ff(r6,a6,0,x,y); //the second transform
g6(x,y)=gg(r6,a6,0,x,y);

r7=.25; //the parameters I want to control for the first transform
a7=0;
xOff7=1;
yOff7=0;

f7(x,y)=ff(r7,a7,0,x,y); //the second transform
g7(x,y)=gg(r7,a7,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,r4,a4,xOff4,yOff4);
controlVar(r5,a5,xOff5,yOff5,r6,a6,xOff6,yOff6,r7,a7,xOff7,yOff7);


iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           f5,xOff5,
           g5,yOff5,
           f6,xOff6,
           g6,yOff6,
           f7,xOff7,
           g7,yOff7,
           "line_horizontal.2d");
```

Is used to generate:two d minkovski1

(go to top)

```
////////////////////Others 1//////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.4; //the parameters I want to control for the first transform
a1=0.7;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.7; //the parameters I want to control for the first transform
a2=-1;
xOff2=0.4;
yOff2=0.35;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.4; //the parameters I want to control for the first transform
a3=0.8;
xOff3=0.95;
yOff3=-0.4;
controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3);
//variable controller

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);


iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           "line_horizontal.2d" );
```

Is used to generate:two_d_minkovski2

(go to top)

```
///////karfiol (Pithagoras tree)///////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.75; //the parameters I want to control for the first transform
a1=.735;
xOff1=0;
```

```
yOff1=0.35;
controlVar(r1, a1,xOff1, yOff1 ); //variable controller

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.5; //the parameters I want to control for the first transform
a2=-.95;
xOff2=0.2;
yOff2=0.5;
controlVar(r2,a2,xOff2,yOff2); //variable controller

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "pentagon.2d" );
```

Is used to generate:two d pithagoras1 two d pithagoras2 two d pithagoras3

(go to top)

```
///////karfiol2 (Pithagoras tree)///////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.75; //the parameters I want to control for the first transform
a1=.785;
xOff1=0;
yOff1=0.33;
controlVar(r1, a1,xOff1, yOff1 ); //variable controller

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.75; //the parameters I want to control for the first transform
a2=-.8;
xOff2=0.2;
yOff2=0.5;
controlVar(r2,a2,xOff2,yOff2); //variable controller

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "pentagon.2d" );
```

Is used to generate:two d pithagoras4

(go to top)

```
///////karfiol3 (Pithagoras net)///////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=1; //the parameters I want to control for the first transform
a1=1.035;
xOff1=0;
yOff1=0.38;
controlVar(r1, a1,xOff1, yOff1 ); //variable controller

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=1; //the parameters I want to control for the first transform
a2=-1.05;
xOff2=0.2;
yOff2=0.65;
controlVar(r2,a2,xOff2,yOff2); //variable controller

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "pentagon.2d" );
```

Is used to generate:two d pithagoras5

(go to top)

```
///////karfiol3 (Pithagoras tree) one more control//////////////
clear();
ff(r,a,b,off,x,y)=r*cos(a)*x - r*sin(b)*y + off; //the generic
transformation
gg(r,a,b,off,x,y)=r*sin(a)*x + r*cos(b)*y + off;

r1=.7; //the parameters I want to control for the first transform
a1=1.89;
b1=-.765;
xOff1=0;
yOff1=0.38;
controlVar(r1, a1, b1, xOff1, yOff1 ); //variable controller

f1(x,y)=ff(r1,a1,b1,0,x,y); //the first transform
```

```
g1(x,y)=gg(r1,a1,b1,0,x,y);

r2=.6; //the parameters I want to control for the first transform
a2=0;
b2=-.8;
xOff2=0.35;
yOff2=0.35;
controlVar(r2,a2,b2,xOff2,yOff2); //variable controller

f2(x,y)=ff(r2,a2,b2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,b2,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           "pentagon.2d" );
```

Is used to generate:two_d_pithagoras6

(go to top)

```
////////////serpinsky triangle////////////////////////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.5; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.5; //the parameters I want to control for the first transform
a2=0;
xOff2=0.2;
yOff2=-0.35;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.5; //the parameters I want to control for the first transform
a3=0;
xOff3=-0.2;
yOff3=-0.35;
controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3);
//variable controller

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);
```

```
iterFract(
           "zIterFractPlugInDemo2",
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           "triangle.2d" );
```

Is used to generate:two d serpinski1 two d serpinski2

(go to top)

```
////////////Serpinski 2/////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.5; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0.23;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.5; //the parameters I want to control for the first transform
a2=0;
xOff2=0.35;
yOff2=-0.5;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.5; //the parameters I want to control for the first transform
a3=0;
xOff3=-0.35;
yOff3=-0.5;
controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3);
//variable controller

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);


iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           "star3.2d" );
```

Is used to generate:two d serpinski3

(go to top)

```
///////////////Serpinski3////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.5; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.5; //the parameters I want to control for the first transform
a2=0;
xOff2=0.16;
yOff2=-0.34;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.5; //the parameters I want to control for the first transform
a3=0;
xOff3=-0.16;
yOff3=-0.34;
controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3);
//variable controller

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);


iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           "star3.2d" );
```

Is used to generate:two d serpinski4

(go to top)

```
///////////serpinski square/////////////////////////////////////////
```

```
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.3; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.3; //the parameters I want to control for the first transform
a2=0;
xOff2=0.25;
yOff2=-.475;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.3; //the parameters I want to control for the first transform
a3=0;
xOff3=0.25;
yOff3=-0.25;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.3; //the parameters I want to control for the first transform
a4=0;
xOff4=-0.25;
yOff4=-0.25;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

r5=.3; //the parameters I want to control for the first transform
a5=0;
xOff5=0.25;
yOff5=0;

f5(x,y)=ff(r5,a5,0,x,y); //the first transform
g5(x,y)=gg(r5,a5,0,x,y);

r6=.3; //the parameters I want to control for the first transform
a6=0;
xOff6=0;
yOff6=-0.475;

f6(x,y)=ff(r6,a6,0,x,y); //the second transform
g6(x,y)=gg(r6,a6,0,x,y);

r7=.3; //the parameters I want to control for the first transform
a7=0;
```

```
xOff7=-0.25;
yOff7=-0.475;

f7(x,y)=ff(r7,a7,0,x,y); //the second transform
g7(x,y)=gg(r7,a7,0,x,y);

r8=.3; //the parameters I want to control for the first transform
a8=0;
xOff8=-0.25;
yOff8=0;

f8(x,y)=ff(r8,a8,0,x,y); //the second transform
g8(x,y)=gg(r8,a8,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,r4,a4,xOff4,yOff4);
//variable controller
controlVar(r5,a5,r6,a6,r7,a7,r8,a8); //variable controller


iterFract(
           "zIterFractPlugInDemo4",
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           f5,xOff5,
           g5,yOff5,
           f6,xOff6,
           g6,yOff6,
           f7,xOff7,
           g7,yOff7,
           f8,xOff8,
           g8,yOff8,
           "triangle.2d" );
```

Is used to generate:two_d_serpinski_sq1

(go to top)

```
///////Spyral///////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.95; //the parameters I want to control for the first transform
a1=.034;
xOff1=0.015;
yOff1=0;
controlVar(r1, a1,xOff1, yOff1 ); //variable controller
```

```
f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

iterFract( f1,xOff1, //first
           g1,yOff1,
           "pentagon.2d");
```

Is used to generate:two d spyral

(go to top)

```
///////////////Torn/////////////////////////
clear();
ff(r,a,off,x,y)=r*cos(a)*x - r*sin(a)*y + off; //the generic transformation
gg(r,a,off,x,y)=r*sin(a)*x + r*cos(a)*y + off;

r1=.45; //the parameters I want to control for the first transform
a1=0;
xOff1=0;
yOff1=0;

f1(x,y)=ff(r1,a1,0,x,y); //the first transform
g1(x,y)=gg(r1,a1,0,x,y);

r2=.45; //the parameters I want to control for the first transform
a2=1.45;
xOff2=0.6;
yOff2=0;

f2(x,y)=ff(r2,a2,0,x,y); //the second transform
g2(x,y)=gg(r2,a2,0,x,y);

r3=.45; //the parameters I want to control for the first transform
a3=-1.5;
xOff3=0.66;
yOff3=0.6;

f3(x,y)=ff(r3,a3,0,x,y); //the second transform
g3(x,y)=gg(r3,a3,0,x,y);

r4=.45; //the parameters I want to control for the first transform
a4=0;
xOff4=0.7;
yOff4=0;

f4(x,y)=ff(r4,a4,0,x,y); //the second transform
g4(x,y)=gg(r4,a4,0,x,y);

controlVar(r1,a1,xOff1,yOff1,r2,a2,xOff2,yOff2,r3,a3,xOff3,yOff3,r4,a4,xOff4,yOff4);

iterFract(
           f1,xOff1, //first
           g1,yOff1,
           f2,xOff2,  //second
           g2,yOff2,
           f3,xOff3,  //third
           g3,yOff3,
           f4,xOff4,  //third
           g4,yOff4,
           "line_horizontal.2d");
```

Is used to generate:two d torn1

(go to top)

""