Fractals using iterated function systems,
with affine transformations
ACCESS
July 2001

```
> restart:#fractals use a lot of memory
> Digits:=4:
    #number of significant digits - this will
    #make computations go faster without sacrificing
    #visual accuracy - because IFS's are self correcting.
> with(plots):
    #we want to be able to see our fractals
Warning, the name changecoords has been redefined
```

The next procedure will take a point P=[x,y]) in the plane and let us compute its image under an affine transformation. We use the same letters for the transformation parameters as we did in the class notes:

$$\mathrm{AFFINE1}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)=\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} e \\ f \end{bmatrix}$$

```
> AFFINE1:=proc(X,a,b,c,d,e,f)
  RETURN(evalf([a*X[1]+b*X[2]+e,
               c*X[1]+d*X[2]+f]));
  end:
>
```

You should check that these are the transformations for the Sierpinski triangle

```
> f1:=P->AFFINE1(P,.5,0,0,.5,.25,.5);
  f2:=P->AFFINE1(P,.5,0,0,.5,.5,0);
  f3:=P->AFFINE1(P,.5,0,0,.5,0,0);
```

$$f1 := P \to \mathrm{AFFINE1}(P, .5, 0, 0, .5, .25, .5)$$

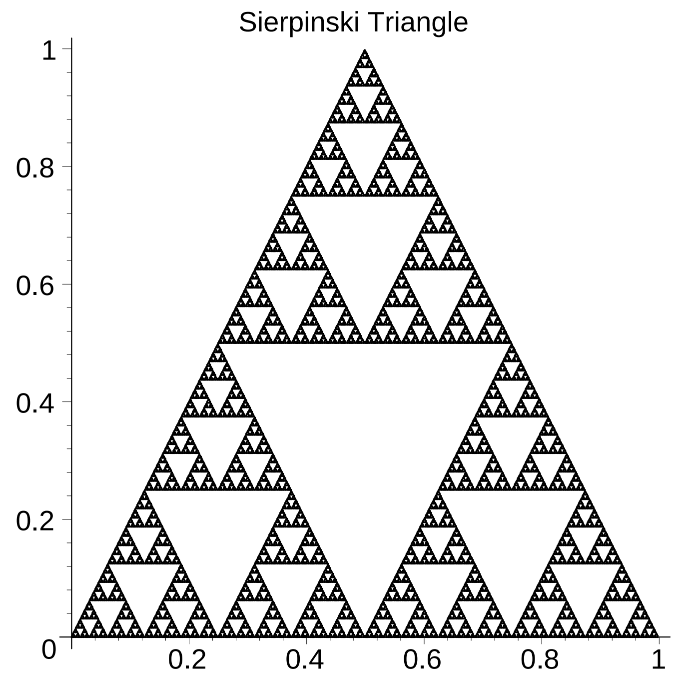
$$f2 := P \to \mathrm{AFFINE1}(P, .5, 0, 0, .5, .5, 0)$$

$$f3 := P \to \mathrm{AFFINE1}(P, .5, 0, 0, .5, 0, 0)$$

```
> S:={[0,0]}:#initial set consisting of one point
> 3^9; #good to keep point numbers below 100,000,
       #because Maple is not the most efficient calculator
```

$$19683$$

```
> for i from 1 to 9 do
  S1:=map(f1,S);
  S2:=map(f2,S);
  S3:=map(f3,S);
  S:=`union`(S1,S2,S3);
  od:
> pointplot(S,symbol=point,scaling=constrained,
      title=`Sierpinski Triangle`);
```




```
>
> restart:
  Digits:=4:
  with(plots):
Warning, the name changecoords has been redefined
> AFFINE1:=proc(X,a,b,c,d,e,f)
  RETURN(evalf([a*X[1]+b*X[2]+e,
                c*X[1]+d*X[2]+f]));
  end:
```

The next procedure let's you use different parameters in specifying your affine map. You can scale the x-direction by r, and rotate it by alpha, then scale the y-direction by s and rotate it by beta. Finally translate by e and f as before: This is the result:

$$\mathrm{AFFINE2}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} r\cos(\alpha) & -s\sin(\beta) \\ r\sin(\alpha) & s\cos(\beta) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

```
> AFFINE2:=proc(X,r,alpha,s,beta,e,f)
  RETURN(AFFINE1(X,r*cos(alpha),-s*sin(beta),
            r*sin(alpha),s*cos(beta),e,f));
  end:
> g1:=P->AFFINE2(P,1/3,0,1/3,0,0,0):
  g2:=P->AFFINE2(P,1/3,Pi/3,1/3,Pi/3,1/3,0):
  g3:=P->AFFINE2(P,1/3,-Pi/3,1/3,-Pi/3,1/2,sqrt(3)/6):
  g4:=P->AFFINE2(P,1/3,0,1/3,0,2/3,0):
```

$$\{[0, 0]\}$$

```
> 4^6;
```

$$4096$$

```
> S:={[0,0]};
```

$$S := \{[0, 0]\}$$

```
> for i from 1 to 6 do
  S1:=map(g1,S);
  S2:=map(g2,S);
  S3:=map(g3,S);
  S4:=map(g4,S);
  S:=`union`(S1,S2,S3,S4);
  od:

>
> pointplot(S,symbol=point,scaling=constrained,
  title=`Koch Snowflake`);
```