This document is written using MAPLE.

In order to make fractals with iterated function systems it is nice to have a test procedure to make sure you have picked your affine maps correctly (and to help you adjust them later if necessary.) The procedure TESTMAP below, takes a list of affine functions as its input, and the result is a mapping picture like the ones in the fractal blueprints from Peitgen’s book.

```maple
> restart; with(plots); Digits := 4:
Warning, the name changecoords has been redefined

> TESTMAP := proc(functlist)  #this procedure lets you test a list of
#functions in your iterated function system
  local num, #the number of functions
          i,  #dummy index
          F,  #current function in list
          S,  #corners of unit square
          L,  #corners of letter L
          Sq,  #unit square
          Llet,  #letter L
          AS,  #transf of square corners
          ASq, #transf of square
          AL,  #transf of L corners
          ALlet,  #transf of letter L
          Pics;  #a list of pictures
   S := [[0, 0], [0, 1], [1, 1], [1, 0]];
   L := [[.1, .9], [.1, .75], [.2, .75], [.2, .775], [.125, .775], [.125, .9]]:
   Sq := polygonplot(S):  #polygonplot connects the dots!
   Llet := polygonplot(L):
   display({Sq, Llet});
   num := nops(functlist):
   for i from 1 to num do
     F := functlist[i]:  #select ith map
     AS[i] := map(F, S):
     AL[i] := map(F, L):
     ASq[i] := polygonplot(AS[i]):
     ALlet[i] := polygonplot(AL[i]):
     #a plot of the transformed square and letter:
     Pics[i] := display({ASq[i], ALlet[i]}):
   od:
   #finally, display the unit square and all its images:
   display({Sq, seq(Pics[i], i = 1 .. num)}, scaling = constrained,
           title = `fractal template');
end:
```

Here is the standard affine map, which encodes
AFFINE1\(\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}\)

\begin{verbatim}
AFFINE1:=proc(X,a,b,c,d,e,f)
RETURN(evalf([a*X[1]+b*X[2]+e,
end:

And in case you want to use it, an alternative version called AFFINE2 which lets you specify scaling factors and rotation angles instead;

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos(\alpha) & -s\sin(\beta) \\ r\sin(\alpha) & s\cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}
\]

\begin{verbatim}
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:

Example 1:

We illustrate the use of TESTMAP with the Sierpinski triangle, and then on the harder twig example (Figure 5.13 in Peitgen’s book.)

\begin{verbatim}
f1:=P->AFFINE1(P,.5,0,0,.5,0,0);
    #shrink by .5 and don’t translate
f2:=P->AFFINE1(P,.5,0,0,.5,.5,0);
    #same shrink, and translate 0.5 to the right
f3:=P->AFFINE1(P,.5,0,0,.5,.25,.5);
    #shrink, then displace by [.25,.5]

> TESTMAP([f1,f2,f3]);
\end{verbatim}
That’s the right blueprint, so now we will generate the fractal, beginning with a single point.

> 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');
Example 2:
We try the twig in Figure 5.13 - since the affine maps weren’t given, I’ve tried parameter values for the matrix and translation until I get a blueprint which looks like the one in the book. Also, I used "restart" (at the top of the file) to clear out memory, and then I re-entered the various procedures I want to use again by using my cursor. Even after the blueprint looked right, it took several tries to get the twig looking good.

```plaintext
> f1:=P->AFFINE1(P,.4,.4,.4,-.4,.24,.55);
f2:=P->AFFINE1(P,.45,.01,.09,.01,-.05,.31);
f3:=P->AFFINE1(P,.45,-.1,0,-.3,.4,.48);

    f1 := P → AFFINE1(P, .4, .4, .4, -.4, .24, .55)
    f2 := P → AFFINE1(P, .45, .01, .09, .01, -.05, .31)
    f3 := P → AFFINE1(P, .45, -.1, 0, -.3, .4, .48)

> TESTMAP([f1,f2,f3]);
```
> S:={[0,0]};

> 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,scaling=constrained,symbol=point,
  title=`twig`,axes=none);