

Willkommen zum Projekt "Die Kochsche Schneeflocke"

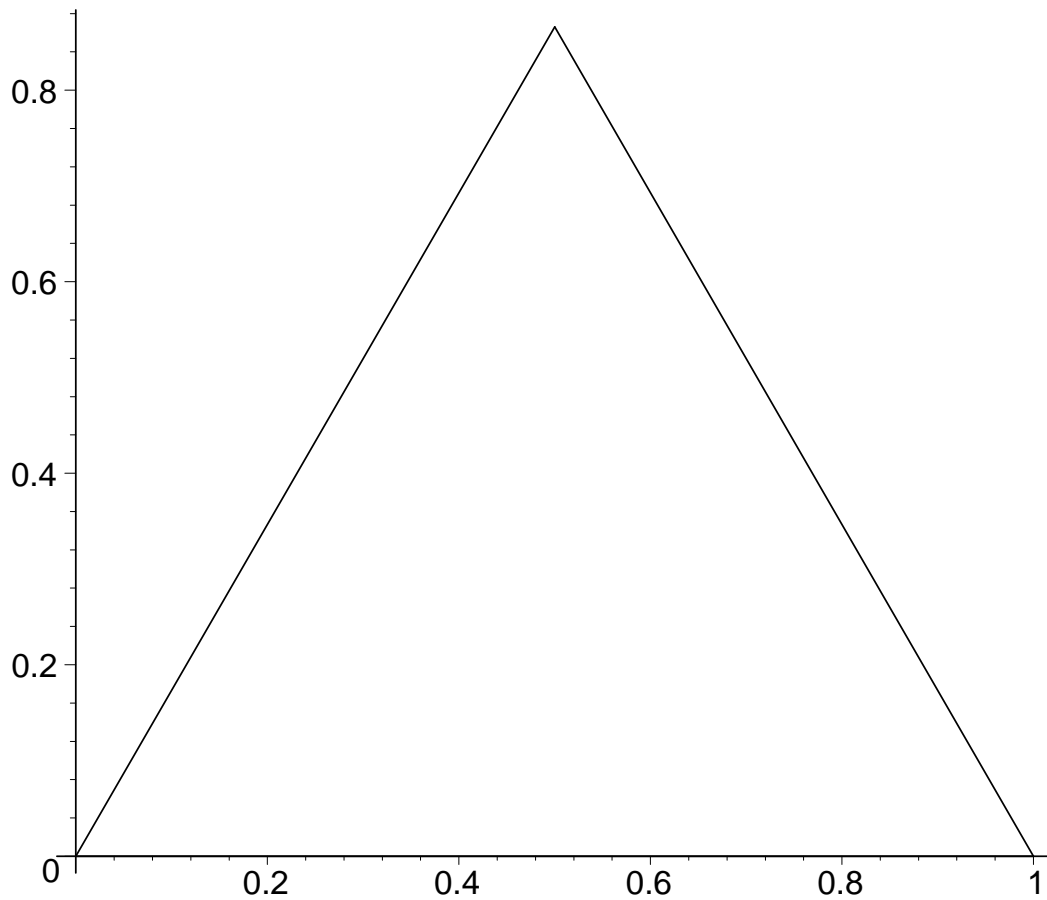
Jan Busse & Friederike Loeser
29.01.2003

Aufgabe: Zeichne die Kochsche Schneeflocke im 0., 1., 2. und 3. Stadium.

Zunächst im 0. Stadium (gezeichnet wird ein *bestimmtes* Dreieck) :

```
restart:
with(plots):
with(plottools):
plot00:=polygon([[0,0],[1,0],[0.5,sqrt(3)/2]]);
plots[display](plot00,scaling=constrained);
Warning, the name changecoords has been redefined
Warning, the name arrow has been redefined

plot00 := POLYGONS([[0., 0.], [1., 0.], [0.5, 0.8660254040]])
```



Um das erste Stadium zu zeichnen berechnen wir zunächst den Schwerpunkt S.

```
A: = <0, 0>;
B: = <1, 0>;
C: = <0.5, sqrt(3)/2>;
S: = (A+B+C)/3;
```

Dann berechnen wir für eine Kante das entstehende Dreieck:

```
plot01:=polygon([[1/3, 0],[1/2, -sqrt(3)/6],[2/3, 0]]):
```

Selbiges drehen wir nun um den Schwerpunkt, so dass an jeder Seite eines entsteht:

```
plot02:=rotate(plot01, 2*Pi/3, convert(S, list)):
plot03:=rotate(plot01, 4*Pi/3, convert(S, list)):
```

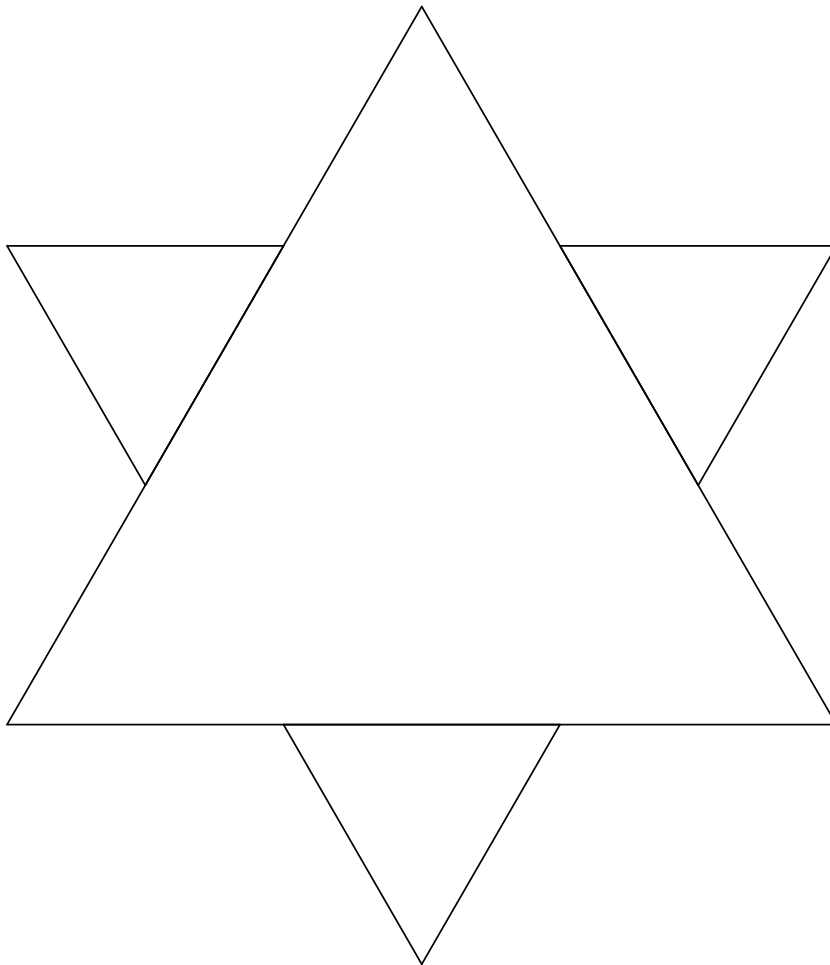
$$A := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C := \begin{bmatrix} 0.5 \\ \frac{\sqrt{3}}{2} \end{bmatrix}$$

$$S := \begin{bmatrix} 0.5000000000 \\ \frac{\sqrt{3}}{6} \end{bmatrix}$$

```
> plots[display](plot00,plot01,plot02,plot03,scaling=constrained,axes=None);
```



[Das gleiche Verfahren wenden wir nun für das zweite Stadium an:

```
> plot04:=polygon([[1/9,0],[1/6,-sqrt(3)/18],[2/9,0]]):  
plot05:=rotate(plot04,Pi/3,convert(S,list)):  
plot06:=rotate(plot04,2*Pi/3,convert(S,list)):  
plot07:=rotate(plot04,Pi,convert(S,list)):  
plot08:=rotate(plot04,4*Pi/3,convert(S,list)):
```

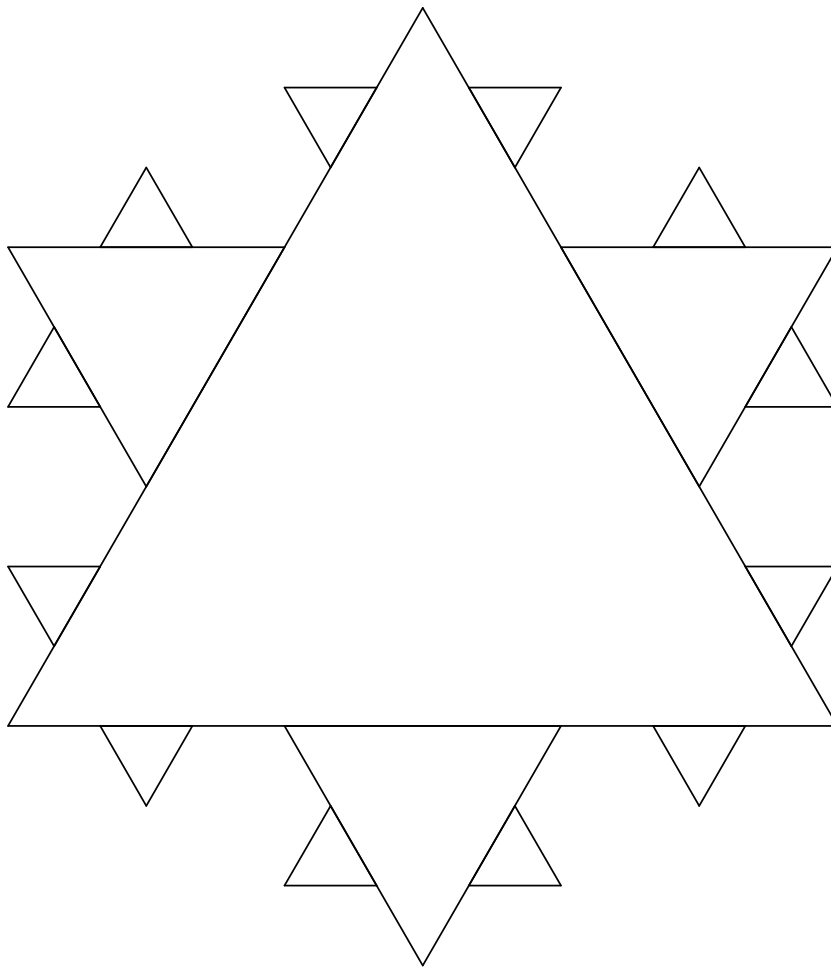
```

plot09:=rotate(plot04,5*Pi/3,convert(S,list)):

plot10:=polygon([[7/9,0],[5/6,-sqrt(3)/18],[8/9,0]]):
plot11:=rotate(plot10,Pi/3,convert(S,list)):
plot12:=rotate(plot10,2*Pi/3,convert(S,list)):
plot13:=rotate(plot10,Pi,convert(S,list)):
plot14:=rotate(plot10,4*Pi/3,convert(S,list)):
plot15:=rotate(plot10,5*Pi/3,convert(S,list)):

> plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,
plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,s
caling=constrained,axes=none);
>

```



[Und auch für das dritte Stadium:

```

> plot16:=polygon([[1/27,0],[1/18,-sqrt(3)/54],[2/27,0]]):
plot17:=rotate(plot16,Pi/3,convert(S,list)):
plot18:=rotate(plot16,2*Pi/3,convert(S,list)):

```

```

plot19:=rotate(plot16,Pi,convert(S,list)):
plot20:=rotate(plot16,4*Pi/3,convert(S,list)):
plot21:=rotate(plot16,5*Pi/3,convert(S,list)):

plot22:=polygon([[7/27,0],[5/18,-sqrt(3)/54],[8/27,0]]):
plot23:=rotate(plot22,Pi/3,convert(S,list)):
plot24:=rotate(plot22,2*Pi/3,convert(S,list)):
plot25:=rotate(plot22,Pi,convert(S,list)):
plot26:=rotate(plot22,4*Pi/3,convert(S,list)):
plot27:=rotate(plot22,5*Pi/3,convert(S,list)):

plot28:=polygon([[19/27,0],[13/18,-sqrt(3)/54],[20/27,0]]):
plot29:=rotate(plot28,Pi/3,convert(S,list)):
plot30:=rotate(plot28,2*Pi/3,convert(S,list)):
plot31:=rotate(plot28,Pi,convert(S,list)):
plot32:=rotate(plot28,4*Pi/3,convert(S,list)):
plot33:=rotate(plot28,5*Pi/3,convert(S,list)):

plot34:=polygon([[25/27,0],[17/18,-sqrt(3)/54],[26/27,0]]):
plot35:=rotate(plot34,Pi/3,convert(S,list)):
plot36:=rotate(plot34,2*Pi/3,convert(S,list)):
plot37:=rotate(plot34,Pi,convert(S,list)):
plot38:=rotate(plot34,4*Pi/3,convert(S,list)):
plot39:=rotate(plot34,5*Pi/3,convert(S,list)):

plot40:=polygon([[10/27,4*sqrt(3)/9],[7/18,25*sqrt(3)/54],[11/27,4*sqrt(3)/9]]):
plot41:=rotate(plot40,Pi/3,convert(S,list)):
plot42:=rotate(plot40,2*Pi/3,convert(S,list)):
plot43:=rotate(plot40,Pi,convert(S,list)):
plot44:=rotate(plot40,4*Pi/3,convert(S,list)):
plot45:=rotate(plot40,5*Pi/3,convert(S,list)):

plot46:=polygon([[16/27,4*sqrt(3)/9],[11/18,25*sqrt(3)/54],[17/27,4*sqrt(3)/9]]):
plot47:=rotate(plot46,Pi/3,convert(S,list)):
plot48:=rotate(plot46,2*Pi/3,convert(S,list)):
plot49:=rotate(plot46,Pi,convert(S,list)):
plot50:=rotate(plot46,4*Pi/3,convert(S,list)):
plot51:=rotate(plot46,5*Pi/3,convert(S,list)):

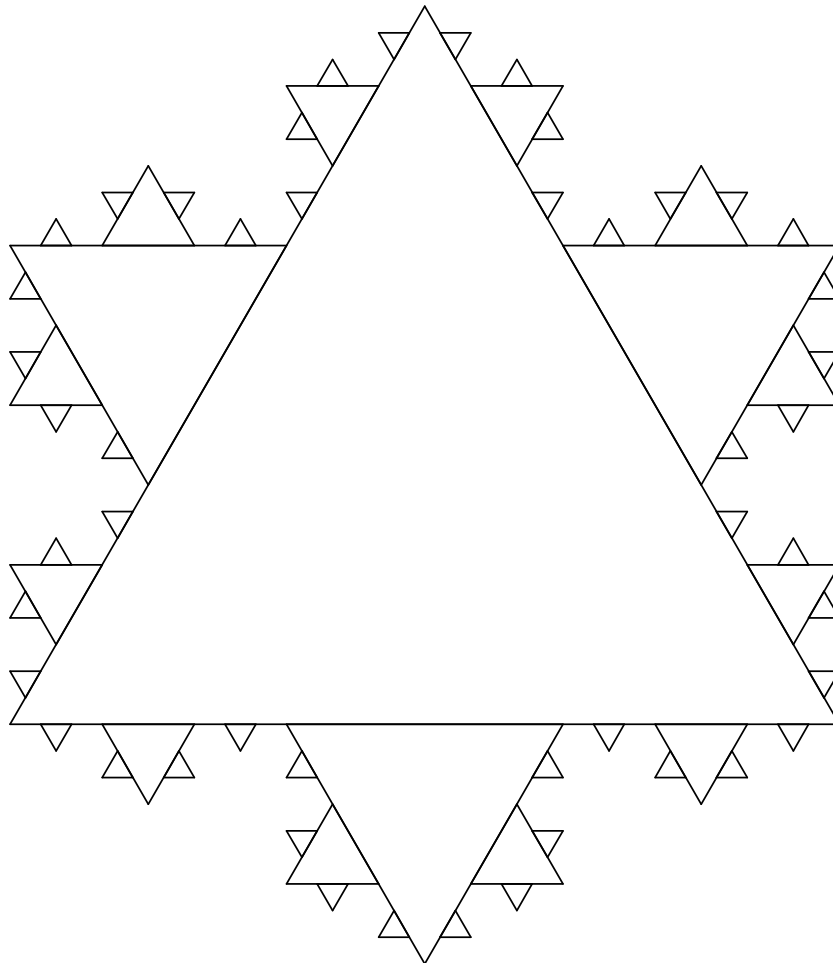
plot52:=polygon([[1/27,2*sqrt(3)/18],[1/18,7*sqrt(3)/54],[2/27,2*sqrt(3)/18]]):
plot53:=rotate(plot52,Pi/3,convert(S,list)):
plot54:=rotate(plot52,2*Pi/3,convert(S,list)):
plot55:=rotate(plot52,Pi,convert(S,list)):
plot56:=rotate(plot52,4*Pi/3,convert(S,list)):
plot57:=rotate(plot52,5*Pi/3,convert(S,list)):

plot58:=polygon([[25/27,2*sqrt(3)/18],[17/18,7*sqrt(3)/54],[26/27,2*sqrt(3)/18]]):
plot59:=rotate(plot58,Pi/3,convert(S,list)):
plot60:=rotate(plot58,2*Pi/3,convert(S,list)):

```

```
plot61:=rotate(plot58,Pi,convert(S,list)):
plot62:=rotate(plot58,4*Pi/3,convert(S,list)):
plot63:=rotate(plot58,5*Pi/3,convert(S,list)):
```

```
> plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,
plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,
plot16,plot17,plot18,plot19,plot20,plot21,plot22,plot23,plot24,
plot25,plot26,plot27,plot28,plot29,plot30,plot31,plot32,plot33,
plot34,plot35,plot36,plot37,plot38,plot39,plot40,plot41,plot42,
plot43,plot44,plot45,plot46,plot47,plot48,plot49,plot50,plot51,
plot52,plot53,plot54,plot55,plot56,plot57,plot58,plot59,plot60,
plot61,plot62,plot63,scaling=constrained,axes=None);
```

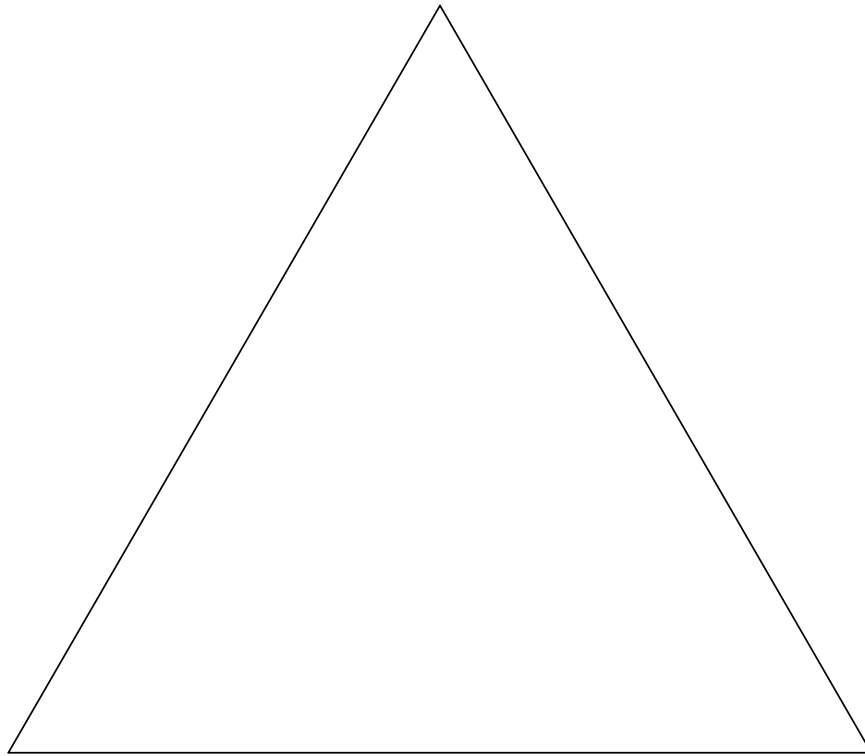


Nun probieren wir einmal unsere bisherigen Ergebnisse in einem Film festzuhalten:

```
p1:=plots[display](plot00,scaling=constrained):
p2:=plots[display](plot00,plot01,plot02,plot03,scaling=constrained):
```

```
p3:=plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,scaling=constrained):
p4:=plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,plot16,plot17,plot18,plot19,plot20,plot21,plot22,plot23,plot24,plot25,plot26,plot27,plot28,plot29,plot30,plot31,plot32,plot33,plot34,plot35,plot36,plot37,plot38,plot39,plot40,plot41,plot42,plot43,plot44,plot45,plot46,plot47,plot48,plot49,plot50,plot51,plot52,plot53,plot54,plot55,plot56,plot57,plot58,plot59,plot60,plot61,plot62,plot63,scaling=constrained):

display([p1,p2,p3,p4],scaling=constrained,insequence=true,axes=none);
```



[So, also ein kleines Erfolgserlebnis;-)!!!

[Nun ist diese Prozedur leider mit einem erheblichen Rechenaufwand verbunden:-(!!!

Probieren wir also etwas neues: Das ganze müsste sich ja auch (gerade für höhere Stufen) in ein Programm einbetten lassen!

Nun also unser Versuch dazu:

```
> restart:
KSF:=proc(punkte,n)
  punkte:= zu betrachtende Punkte
  n:= (restliche) Rekursionstiefe
  local i,j,Mi,P,neuepunkte,Liste;
  Liste := <[polygon([punkte[2],punkte[3],punkte[4]])]>;
  if (n>0) then
    for j to 4 do
      (4 Strecken sind in "punkte")
```

```

    if (punkte[j] <> punkte[j+1]) then
      (Gleichheit tritt nur im ersten Fall ein, da hier ein Punkt doppelt vorkommt: p3,p3)
      P := punkte[j+1]-punkte[j];
      (P:= Vektor zwischen p[j] und p[j+1])
      Mi := evalf(punkte[j]+1/2*P +
(1/2)*(1/sqrt(3))*[-P[2],P[1]]);
      (Mi:= 3/ mittlerer Punkt der Unterteilung einer Strecke in 4
Teilstrecken)
      neuepunkte :=
[punkte[j],punkte[j]+1/3*P,Mi,punkte[j]+2/3*P,punkte[j+1]];
      (neuepunkte: die 4 neuen Strecken / 5 neuen Punkte über der alten
Strecke)
      Liste := <Liste,KSF(neuepunkte,n-1)>;
    end if;
  end do;
end if;
return Liste;
end proc;

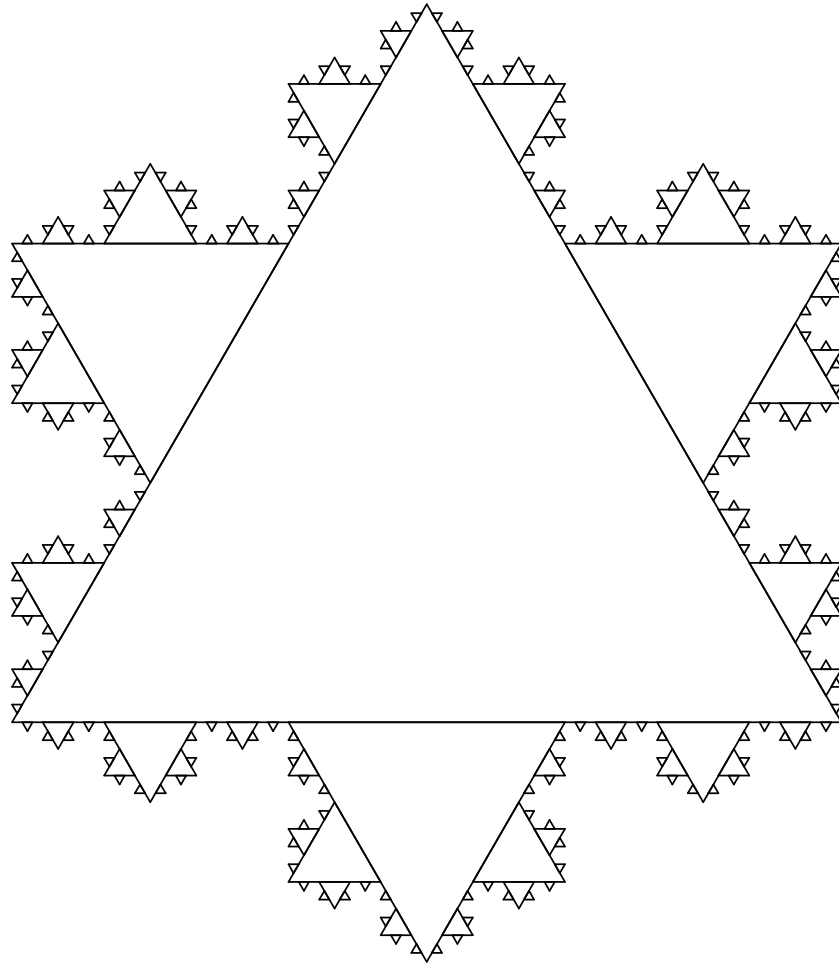
```

Nun haben wir ein kleines Programm entwickelt, das die Stufen bzw. äußeren "Enden" berechnet. Fehlt noch eine Prozedur, die das ganze auch ersichtlich anwendet (u.a. bezogen auf spezielle Punkte):

```

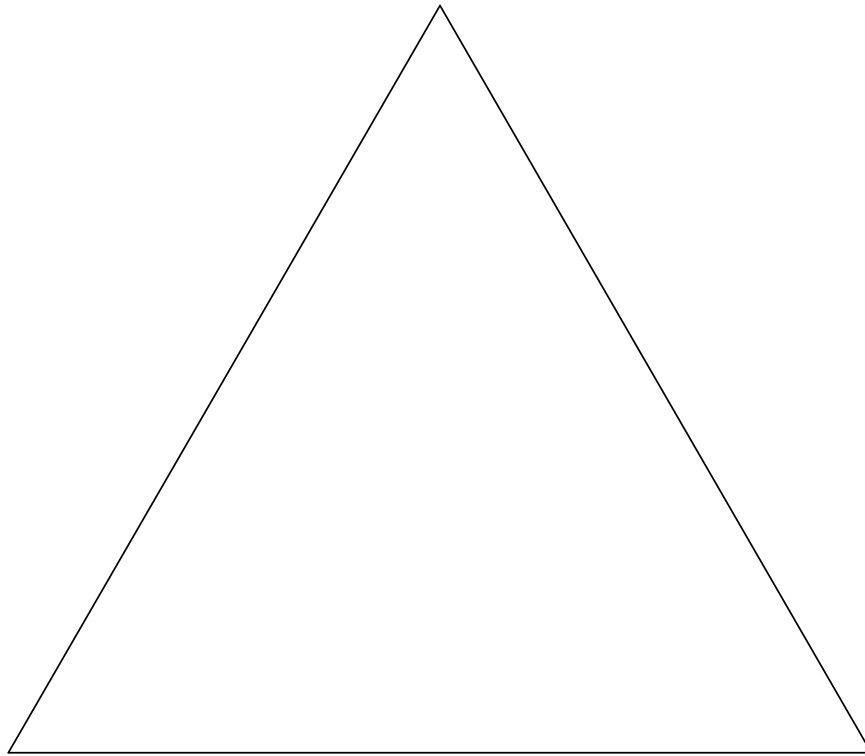
>
> KSFFULL:= proc(n)
  local L,p1,p2,p3,a1;
  p1 := [0,0]:
  p2 := [1,0]:
  p3 := [1/2,1/2*sqrt(3)]:
  a1 := [p3,p3,p2,p1,p3]:
  if (n>=1) then
    L := KSF(a1,n);
    with(plottools):
plots[display]('L[k,1]'$k=1..2^(2*n),scaling=constrained,axes=no
ne);
  else
    with(plottools):
plots[display](polygon([p1,p2,p3]),scaling=constrained,axes=none
);
  end if;
end proc;
>
> KSFFULL(4);

```



Und nun das Projekt "Film" noch einmal ein bißchen besser;-)

```
> plots[display](KSFFULL(0),KSFFULL(1),KSFFULL(2),KSFFULL(3),KSFFULL(4),KSFFULL(5),insequence=true, scaling=constrained);
```



[>

Längenberechnung nach dem Riemannschen Ansatz

Welche Länge hat die Kochsche Schneeflocke?

Wir wollen dazu jede Seite einzeln betrachten. Die Stufe in der wir uns befinden soll n sein. Jede Seite besteht somit aus 4^n Strecken, die jeweils die Länge $1/(3^n)$ haben.

Somit hat eine Kante nach n Stufen die Länge

[> `restart;`
`Länge:=4^n*(1/(3^n));`

$$Länge = \frac{4^n}{3^n}$$

[Die gesamte Länge der Kochschen Schneeflocke beträgt somit:

[> `LängeKSF:=3*Länge;`

$$LängeKSF = \frac{3 \cdot 4^n}{3^n}$$

Um nun die Länge der Kochschen Schneeflocke für möglichst kleine Strecken zu betrachten lassen wir die Stufen n gegen unendlich laufen.

```
limit(LängeKSF,n=infinity);
```

∞

Nun vergleichen wir die Länge mit der Fläche, die entsteht, wenn man die einzelnen Streckenteile a Quadrate auffasst. Somit gilt für die Fläche die folgende Formel:

```
> FlächeKSF:=3*4^n*(1/(3^n)^2);
```

$$FlächeKSF = \frac{3 \cdot 4^n}{(3^n)^2}$$

Wieder bezogen auf möglichst kleine Strecken, also die höchste Stufe der Entwicklung gilt:

```
limit(FlächeKSF,n=infinity);
```

0

>

Nun erscheint die Frage interessant, wann der Sprung von "unendlich" auf "0" geht! Dazu nehmen wir eine Prozedur, die die ("triviale") Dimension (=: x) beim Sprung auf Null annähert:

```
> restart:
```

```
g:=Genauigkeit
```

```
S:=Sprung
```

```
Dim:=proc(g)
```

```
local S,x:
```

```
for x by 1/g to 3 do
```

```
S:=limit(4^n*((1/3)^n)^x,n=infinity):
```

```
if S=0 then return x-1/(2*g):
```

```
(Hier gehen wir einen "kleinen Schritt" vor dem Sprung zurück...)
```

```
end if:
```

```
end do:
```

```
return "kein x gefunden":
```

```
end proc:
```

```
> evalf(Dim(100));
```

1.265000000

```
> evalf(Dim(1000));
```

1.260500000

>

>

>

Da die Kochsche Schneeflocke als Fraktal angesehen wird, kann man auch ihre fraktale Dimension berechnen:

Die Definition der Fraktalen Dimension bezieht sich auf das Verhältnis zwischen der ursprünglich Länge und der neuen Länge. Dabei ist zu beachten, dass die neue Strecke unabhängig davon ist, ob aus mehreren einzelnen Strecken besteht oder ob sie eine "gerade" Strecke bildet.

Das Verhältnis wird mit

$$S_{neu} = S_{alt} k^d$$

bezeichnet. Dabei ist "k" der Streckfaktor (im Fall der Kochschen Schneeflocke ist $k=3$).

Betrachten wir nun die Strecken: Die alte Strecke sei nun eine Einheit lang. Dementsprechend ist die neue Strecke 4 Einheiten lang, da man die alte Strecke vier mal in der neuen wiederfindet.

Die fraktale Dimension wird mit "d" benannt.

Somit ergibt sich die folgende fraktale Dimension:

```
> d:=evalf(log(4)/log(3));  
d := 1.261859507
```

Interessant ist, dass diese Form der Dimension fasst mit dem Sprung bei der "trivialen" Dimension übereinstimmt....?!

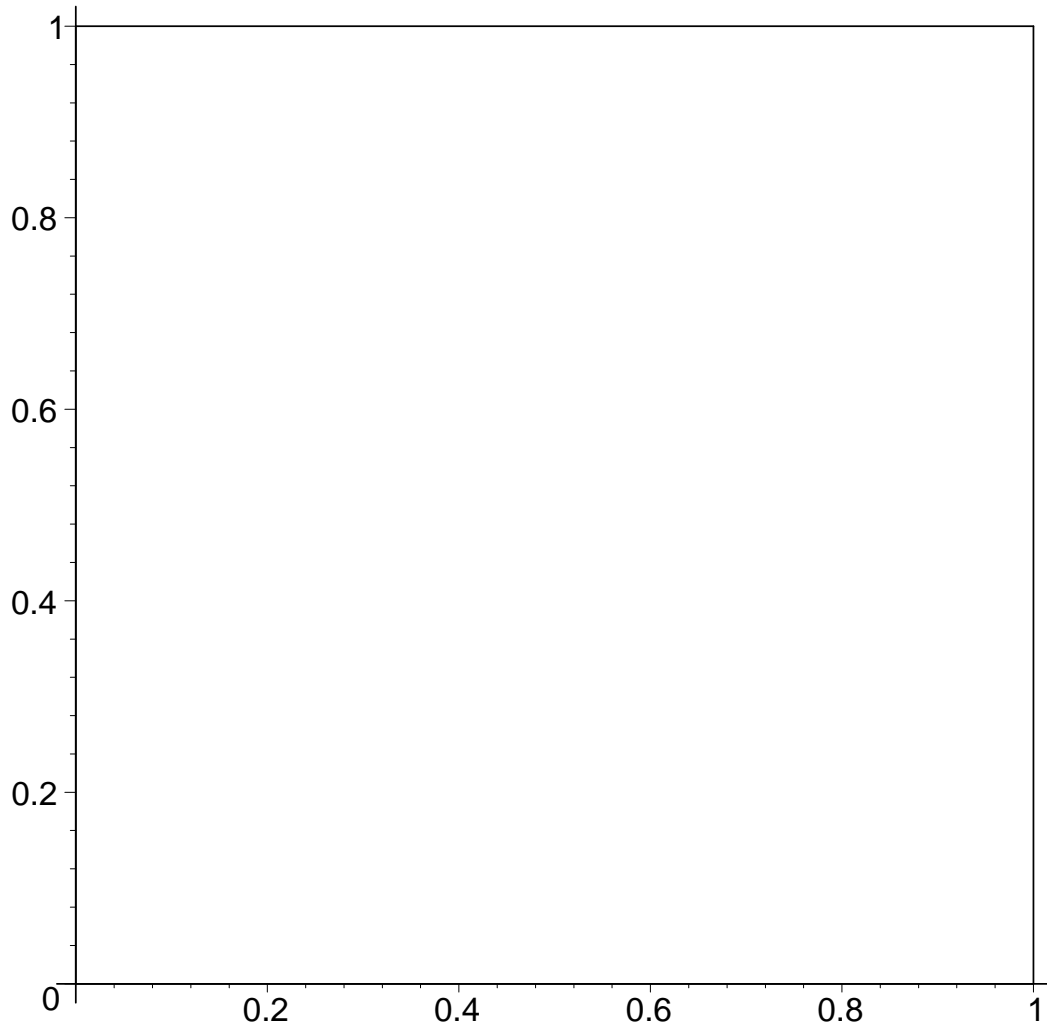
Hat man sich nun genug mit diesem kleinen Problem auseinandergesetzt, warten natürlich noch weitere auf uns!

Wie wäre es zum Beispiel mit folgender Figur? Einem Quadrat, dessen Seiten in fünf Teile aufgeteilt werden. Über dem zweitem und viertem Abschnitt lassen wir ein neues Quadrat der Kantenlänge $1/5$ der Ausgangskantenlänge entstehen...

(Die Art und Weise der Programmierung ist wie bei der ersten Aufgabe...)

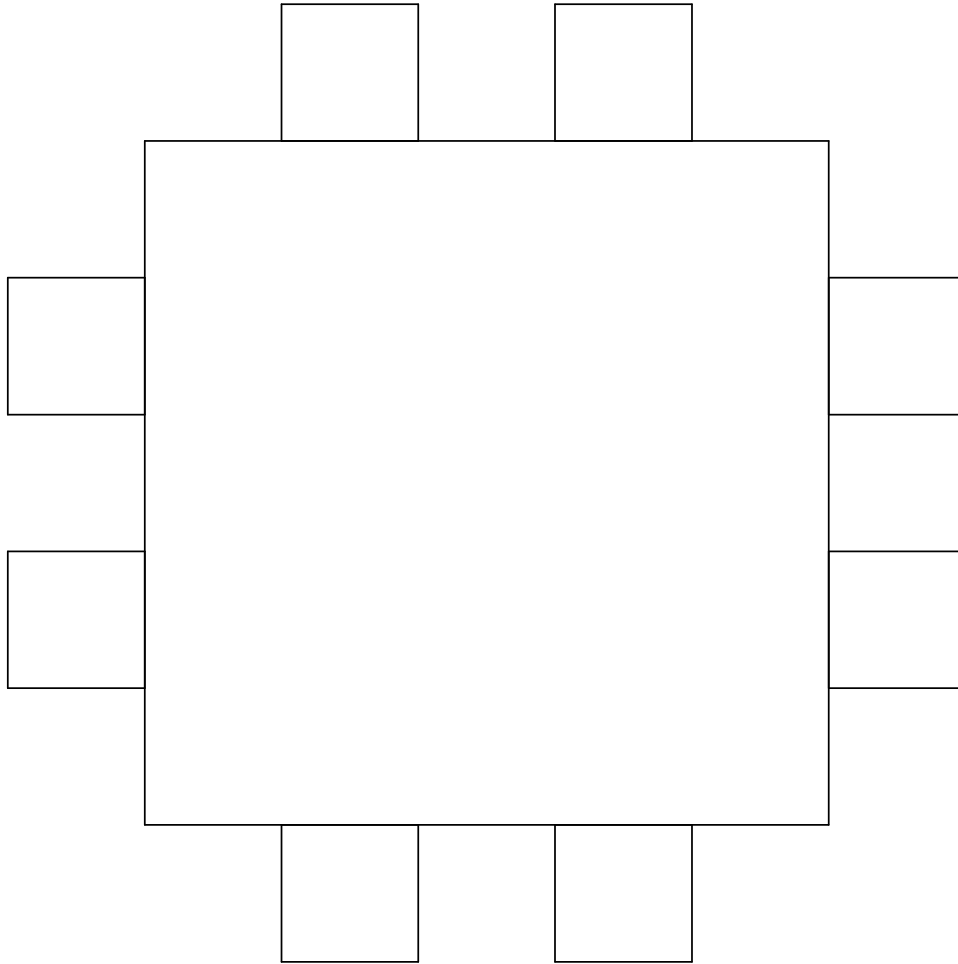
Also: Lust auf einen Flickenteppich?

```
>  
> with(plottools):  
with(plots):  
Die nullte Stufe:  
Q:=[[0,0],[0,1],[1,1],[1,0]]:  
plot0:=polygonplot(Q,scaling=constrained):  
plots[display](plot0,scaling=constrained);  
Warning, the names arrow and changecoords have been redefined
```



Und zur ersten Stufe:

```
S:=[0.5,0.5]:  
plot01:=polygonplot([[0.2,0],[0.4,0],[0.4,-0.2],[0.2,-0.2]]):  
plot02:=rotate(plot01,Pi/2,convert(S,list)):  
plot03:=rotate(plot01,Pi,convert(S,list)):  
plot04:=rotate(plot01,3*Pi/2,convert(S,list)):  
  
plot05:=polygonplot([[0.6,0],[0.8,0],[0.8,-0.2],[0.6,-0.2]]):  
plot06:=rotate(plot05,Pi/2,convert(S,list)):  
plot07:=rotate(plot05,Pi,convert(S,list)):  
plot08:=rotate(plot05,3*Pi/2,convert(S,list)):  
  
> plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,  
plot07,plot08,scaling=constrained,axes=none);
```



Und, last but not least, die dritte Sufo:

```
plot09:=polygonplot([[0.04,0],[0.08,0],[0.08,-0.04],[0.04,-0.04]])
:
plot10:=rotate(plot09,Pi/2,convert(S,list)):
plot11:=rotate(plot09,Pi,convert(S,list)):
plot12:=rotate(plot09,3*Pi/2,convert(S,list)):

plot13:=polygonplot([[0.12,0],[0.16,0],[0.16,-0.04],[0.12,-0.04]])
:
plot14:=rotate(plot13,Pi/2,convert(S,list)):
plot15:=rotate(plot13,Pi,convert(S,list)):
plot16:=rotate(plot13,3*Pi/2,convert(S,list)):

plot17:=polygonplot([[0.2,-0.04],[0.16,-0.04],[0.16,-0.08],[0.2,-0.08]])
:
plot18:=rotate(plot17,Pi/2,convert(S,list)):
plot19:=rotate(plot17,Pi,convert(S,list)):
```

```
plot20:=rotate(plot17,3*Pi/2,convert(S,list)):

plot21:=polygonplot([[0.2,-0.12],[0.16,-0.12],[0.16,-0.16],[0.2,-0.16]]):
plot22:=rotate(plot21,Pi/2,convert(S,list)):
plot23:=rotate(plot21,Pi,convert(S,list)):
plot24:=rotate(plot21,3*Pi/2,convert(S,list)):

plot25:=polygonplot([[0.24,-0.2],[0.28,-0.2],[0.28,-0.24],[0.24,-0.24]]):
plot26:=rotate(plot25,Pi/2,convert(S,list)):
plot27:=rotate(plot25,Pi,convert(S,list)):
plot28:=rotate(plot25,3*Pi/2,convert(S,list)):

plot29:=polygonplot([[0.32,-0.2],[0.36,-0.2],[0.36,-0.24],[0.32,-0.24]]):
plot30:=rotate(plot29,Pi/2,convert(S,list)):
plot31:=rotate(plot29,Pi,convert(S,list)):
plot32:=rotate(plot29,3*Pi/2,convert(S,list)):

plot33:=polygonplot([[0.4,-0.12],[0.4,-0.16],[0.44,-0.16],[0.44,-0.12]]):
plot34:=rotate(plot33,Pi/2,convert(S,list)):
plot35:=rotate(plot33,Pi,convert(S,list)):
plot36:=rotate(plot33,3*Pi/2,convert(S,list)):

plot37:=polygonplot([[0.4,-0.04],[0.4,-0.08],[0.44,-0.08],[0.44,-0.04]]):
plot38:=rotate(plot37,Pi/2,convert(S,list)):
plot39:=rotate(plot37,Pi,convert(S,list)):
plot40:=rotate(plot37,3*Pi/2,convert(S,list)):

plot41:=polygonplot([[0.44,0],[0.48,0],[0.48,-0.04],[0.44,-0.04]])
:
plot42:=rotate(plot41,Pi/2,convert(S,list)):
plot43:=rotate(plot41,Pi,convert(S,list)):
plot44:=rotate(plot41,3*Pi/2,convert(S,list)):

plot45:=polygonplot([[0.52,0],[0.56,0],[0.56,-0.04],[0.52,-0.04]])
:
plot46:=rotate(plot45,Pi/2,convert(S,list)):
plot47:=rotate(plot45,Pi,convert(S,list)):
plot48:=rotate(plot45,3*Pi/2,convert(S,list)):

plot49:=polygonplot([[0.6,-0.04],[0.6,-0.08],[0.56,-0.08],[0.56,-0.04]]):
plot50:=rotate(plot49,Pi/2,convert(S,list)):
plot51:=rotate(plot49,Pi,convert(S,list)):
plot52:=rotate(plot49,3*Pi/2,convert(S,list)):

plot53:=polygonplot([[0.6,-0.12],[0.6,-0.16],[0.56,-0.16],[0.56,-0.12]]):
plot54:=rotate(plot53,Pi/2,convert(S,list)):
```

```

plot55:=rotate(plot53,Pi,convert(S,list)):
plot56:=rotate(plot53,3*Pi/2,convert(S,list)):

plot57:=polygonplot([[0.64,-0.2],[0.68,-0.2],[0.68,-0.24],[0.64,-0.24]]):
plot58:=rotate(plot57,Pi/2,convert(S,list)):
plot59:=rotate(plot57,Pi,convert(S,list)):
plot60:=rotate(plot57,3*Pi/2,convert(S,list)):

plot61:=polygonplot([[0.72,-0.2],[0.76,-0.2],[0.76,-0.24],[0.72,-0.24]]):
plot62:=rotate(plot61,Pi/2,convert(S,list)):
plot63:=rotate(plot61,Pi,convert(S,list)):
plot64:=rotate(plot61,3*Pi/2,convert(S,list)):

plot65:=polygonplot([[0.8,-0.12],[0.8,-0.16],[0.84,-0.16],[0.84,-0.12]]):
plot66:=rotate(plot65,Pi/2,convert(S,list)):
plot67:=rotate(plot65,Pi,convert(S,list)):
plot68:=rotate(plot65,3*Pi/2,convert(S,list)):

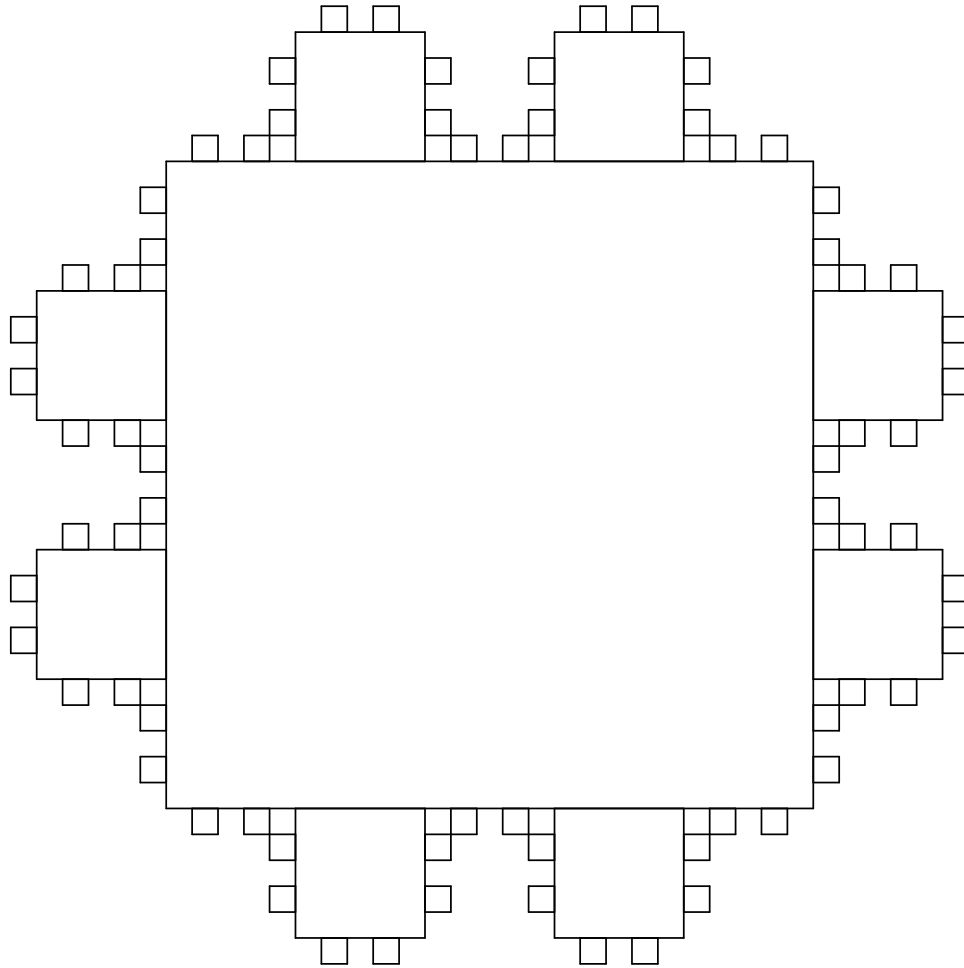
plot69:=polygonplot([[0.8,-0.04],[0.8,-0.08],[0.84,-0.08],[0.84,-0.04]]):
plot70:=rotate(plot69,Pi/2,convert(S,list)):
plot71:=rotate(plot69,Pi,convert(S,list)):
plot72:=rotate(plot69,3*Pi/2,convert(S,list)):

plot73:=polygonplot([[0.84,0],[0.88,0],[0.88,-0.04],[0.84,-0.04]]):
:
plot74:=rotate(plot73,Pi/2,convert(S,list)):
plot75:=rotate(plot73,Pi,convert(S,list)):
plot76:=rotate(plot73,3*Pi/2,convert(S,list)):

plot77:=polygonplot([[0.92,0],[0.96,0],[0.96,-0.04],[0.92,-0.04]]):
:
plot78:=rotate(plot77,Pi/2,convert(S,list)):
plot79:=rotate(plot77,Pi,convert(S,list)):
plot80:=rotate(plot77,3*Pi/2,convert(S,list)):

plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,plot16,plot17,plot18,plot19,plot20,plot21,plot22,plot23,plot24,plot25,plot26,plot27,plot28,plot29,plot30,plot31,plot32,plot33,plot34,plot35,plot36,plot37,plot38,plot39,plot40,plot41,plot42,plot43,plot44,plot45,plot46,plot47,plot48,plot49,plot50,plot51,plot52,plot53,plot54,plot55,plot56,plot57,plot58,plot59,plot60,plot61,plot62,plot63,plot64,plot65,plot66,plot67,plot68,plot69,plot70,plot71,plot72,plot73,plot74,plot75,plot76,plot77,plot78,plot79,plot80,scaling=constrained,axes=none);

```

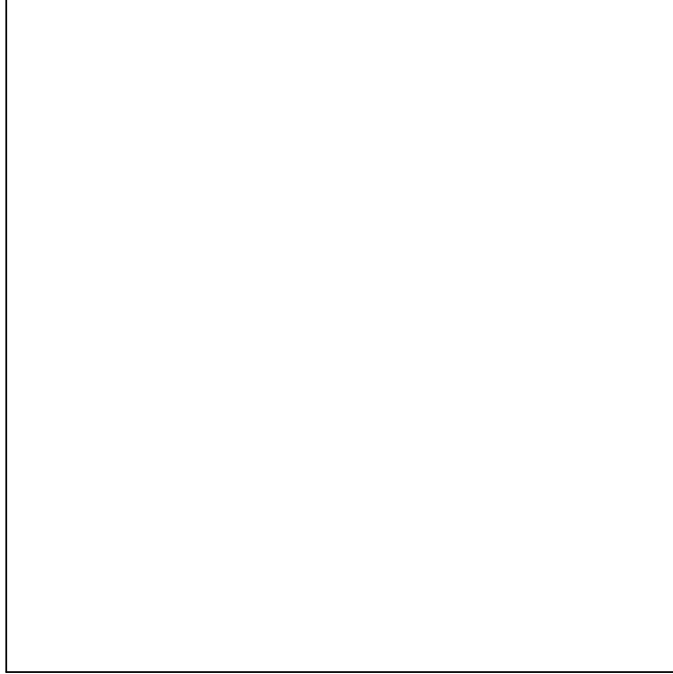


[>

Und nun (für alle Cineasten) noch mal als Filmchen...

```
[ > p1:=plots[display](plot00,scaling=constrained):  
[ > p2:=plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,plot07,plot08,scaling=constrained):  
[ > p3:=plots[display](plot00,plot01,plot02,plot03,plot04,plot05,plot06,plot07,plot08,plot09,plot10,plot11,plot12,plot13,plot14,plot15,plot16,plot17,plot18,plot19,plot20,plot21,plot22,plot23,plot24,plot25,plot26,plot27,plot28,plot29,plot30,plot31,plot32,plot33,plot34,plot35,plot36,plot37,plot38,plot39,plot40,plot41,plot42,plot43,plot44,plot45,plot46,plot47,plot48,plot49,plot50,plot51,plot52,plot53,plot54,plot55,plot56,plot57,plot58,plot59,plot60,plot61,plot62,plot63,plot64,plot65,plot66,plot67,plot68,plot69,plot70,plot71,plot72,plot73,plot74,plot75,plot76,plot77,plot78,plot79,plot80,scaling=constrained):  
[ > plots[display](p1,p2,p3,scaling=constrained,insequence=true,axes
```

```
=none);
```



```
[ >
```

Als Abschlussbonus könne wir nun ja noch einmal die fraktale Dimension des "Flickens" berechnen:

Die neue Länge ist das fünffache von der ersten und setzt sich aus neun Teilstücken zusammen. Als folgt die fraktale Dimension der Größe...

```
[ >
```

```
d:=evalf(log(9)/log(5));
```

```
d := 1.365212389
```

```
[ > restart:
```

```
g:=Genauigkeit
```

```
S:=Sprung
```

```
Dim:=proc(g)
```

```
local S,x:
```

```
for x by 1/g to 3 do
S:=limit(9^n*((1/5)^n)^x,n=infinity):
if S=0 then return x-1/(2*g):
#(Hier gehen wir einen "kleinen Schritt" vor dem Sprung zurück...)
end if:
end do:
return "kein x gefunden":
end proc:
```

```
> evalf(Dim(100));
```

1.365000000

```
>
```

```
Und wieder ist alles so, wie es sein sollte...
```

Und nun: aufwachen!
Tschüss!!!

```
>
```