1. TI Draw Basics

De module TI Draw bevat een aantal functies(methodes) die het toelaat om segmenten, rechthoeken, veelhoeken, cirkels, bogen en tekst toe te voegen aan het grafische venster van de shell.

De default-configuratie heeft (0,0) als linkerbovenhoek met de positieve richting van de x-as naar rechts en de positieve richting van de y-as naar beneden.

```
from ti_draw import *
draw_text(2,16,"(0,0)")
draw_text(262,210,"(318,212)")
```

Deze configuratie is niet zo vertrouwelijk als de oriëntatie van een standaard wiskundig assenstelsel. Hiervoor veranderen we de oriëntatie van het y-as als volgt:

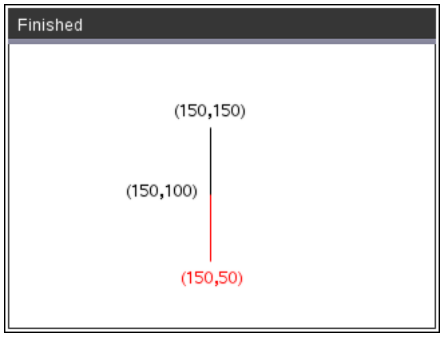
```
from ti_draw import *
dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])
draw_text(2,2,"(0,0)")
draw_text(262,195,"(318,212)")
```

We verduidelijken de beschikbare functies voor het tekenen van wiskundige figuren.

1.1. Segmenten

draw_line(x1,y1,x2,y2) tekent een segment tussen de punten (x1,y1) en (x2,y2).

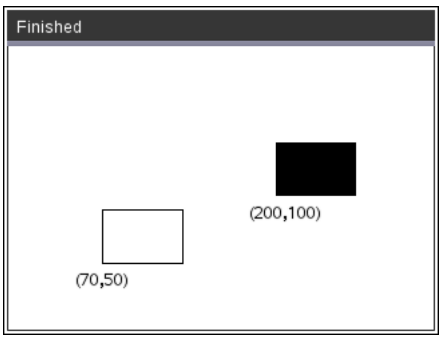
```
from ti_draw import *
dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])
draw_line(150,100,150,150)
draw_text(87,95,"(150,100)")
draw_text(123,155,"(150,150)")
set_color(255,0,0)
draw_line(150,100,150,50)
draw_text(128,30,"(150,50)")
```



1.2. Rechthoeken

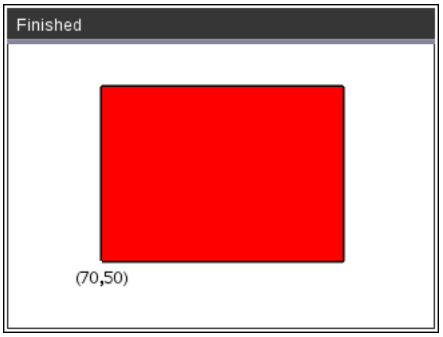
draw_rect(x,y,breedte,hoogte) – fill_rect(x,y,breedte,hoogte) tekent een rechthoek zoals hieronder aangegeven met (x,y) als linkerbenedenhoek.

```
from ti_draw import *
dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])
draw_rect(70,50,60,40)
draw_text(50,30,"(70,50)")
fill_rect(200,100,60,40)
draw_text(180,80,"(200,100)")
```



Een rechthoek kan ook getekend worden als veelhoek d.m.v lijsten met de coördinaten van de hoekpunten van de rechthoek.

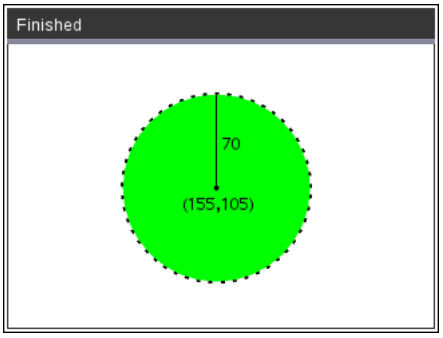
```
from ti_draw import *
dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])
draw_text(50,30,"(70,50)")
xcoord=[70,250,250,70,70]
ycoord=[50,50,180,180,50]
set_pen("medium","solid")
draw_poly(xcoord,ycoord)
set_color(255,0,0)
fill_poly(xcoord,ycoord)
```



1.3. Cirkels

draw_circle(x,y,straal) – fill_circle(x,y,straal) tekent een cirkel met middelpunt (x,y).

```
from ti_draw import *
dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])
set_pen("medium","dotted")
draw_circle(155,105,70)
set_color(0,255,0)
fill_circle(155,105,70)
set_color(0,0,0)
fill_circle(155,105,2)
draw_text(130,85,"(155,105)")
set_pen("thin","solid")
draw_text(160,130,"70")
draw_line(155,105,155,175)
```

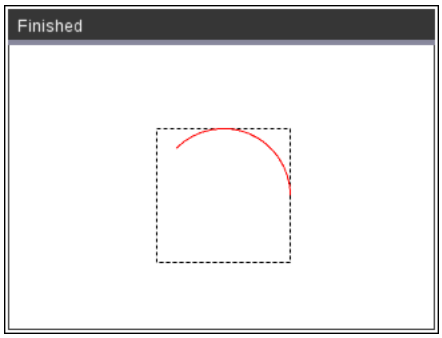


1.4. Cirkelbogen

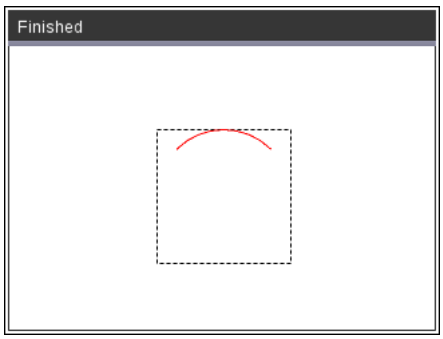
draw_arc(x,y,breedte,hoogte,beginhoek,hoekgrootte) – fill_arc(x,y,breedte,hoogte,beginhoek,hoekgrootte)

Deze functie tekent een boog op de cirkel ingesloten in de rechthoek, draw_rect(x,y,breedte,hoogte), waarbij (x,y) het linkerbenedenhoekpunt is t.o.v. het door ons gedefinieerde assenstelsel/window.

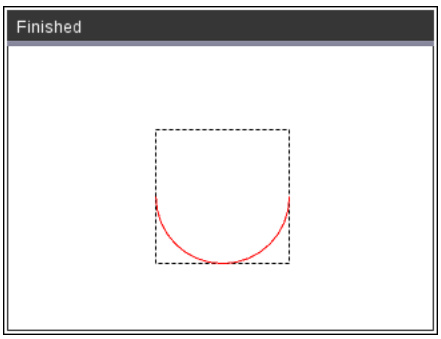
```
from ti_draw import *

dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])

h=100
set_pen("thin","dashed")
draw_rect(110,50,h,h)
set_color(255,0,0)
set_pen("thin","solid")
draw_arc(110,50,h,h,0,135)
```



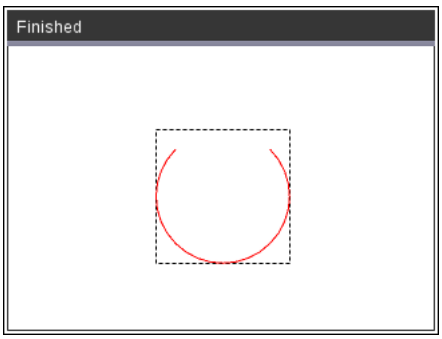
Nog enkele voorbeelden



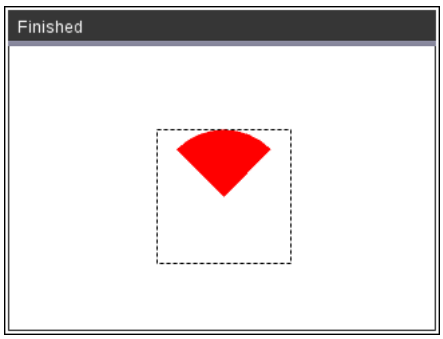
draw_arc(110,50,100,100,45,90)



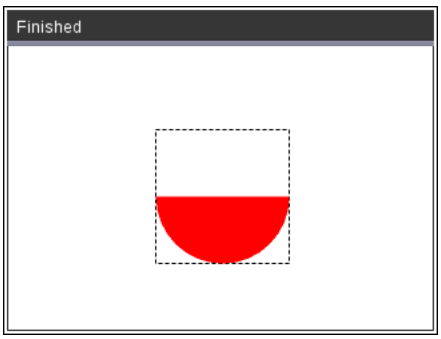
draw_arc(110,50,100,100,180,180)



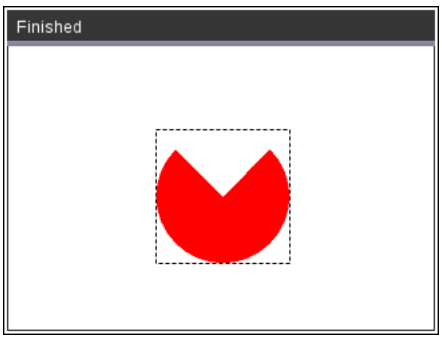
draw_arc(110,50,100,100,135,270)



draw_arc(110,50,100,100,45,90)



draw_arc(110,50,100,100,180,180)



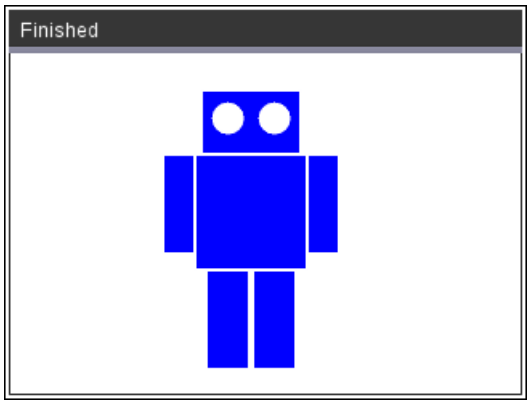
draw_arc(110,50,100,100,135,270)

2. Creatief met lijnen en bogen

2.1. Robot

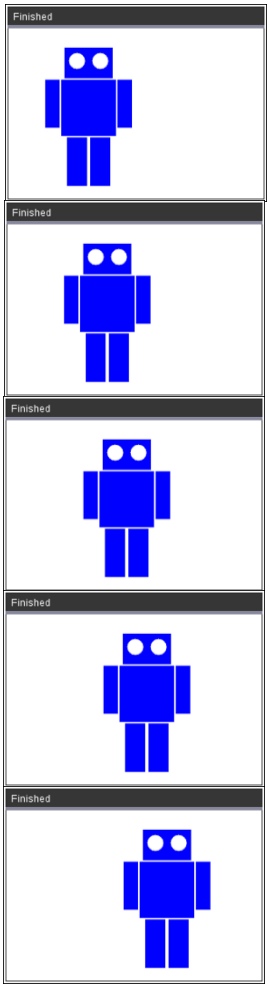
Met enkele rechthoeken en twee cirkels tekenen we onderstaande robot:

```
from ti_draw import *

dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])

set_color(0,0,255)
fill_rect(120,150,60,38)
fill_rect(116,78,68,70)
fill_rect(96,88,18,60)
fill_rect(186,88,18,60)
fill_rect(123,16,25,60)
fill_rect(152,16,25,60)
set_color(255,255,255)
fill_circle(135,172,10)
fill_circle(164,172,10)
```



Het tekenen van de robot in een for-loop plaatsen,
zet de robot in beweging:

```
from ti_draw import *

dim=get_screen_dim()
set_window(0,dim[0],0,dim[1])

use_buffer()

for i in range(-50,50):
    clear()
    set_color(0,0,255)
    fill_rect(120+i,150,60,38)
    fill_rect(116+i,78,68,70)
    fill_rect(96+i,88,18,60)
    fill_rect(186+i,88,18,60)
    fill_rect(123+i,16,25,60)
    fill_rect(152+i,16,25,60)
    set_color(255,255,255)
    fill_circle(135+i,172,10)
    fill_circle(164+i,172,10)
    paint_buffer()
```



Zonder gebruik te maken van de buffer()-statements, genereert
de for-lus een flikkerend beeld van de bewegende robot.

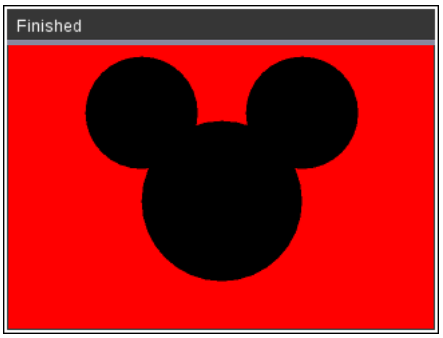
use_buffer() zorgtdat alles in de achtergrond(geheugen) wordt getekend totdat
paint_buffer() de buffer tekent.

Het is aan te raden de buffer te gebruiken indien veel objecten getekend worden (snelheid) en indien het scherm regelmatig wordt gewist voor nieuwe objecten (flikkeren).

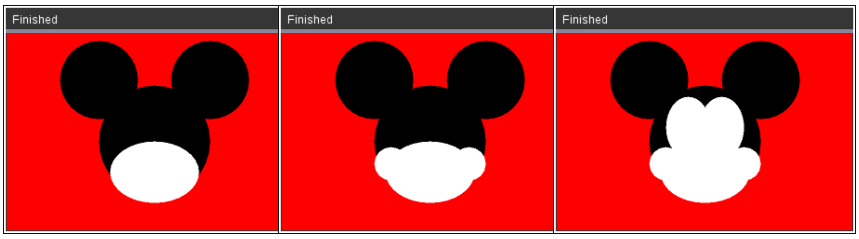
2.2. Mickey

Voor het tekenen van Mickey configureren we het scherm met de oorsprong in het midden:

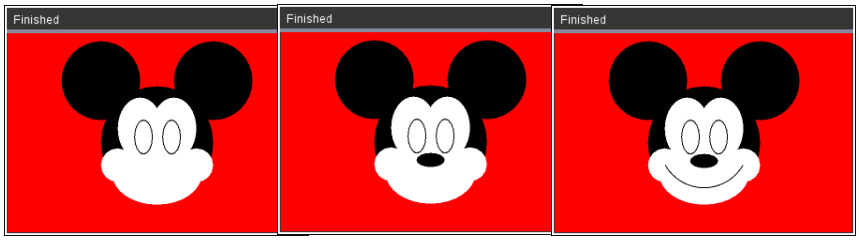
```
from ti_draw import *
dim=get_screen_dim()
set_window(-dim[0]/2,dim[0]/2,-dim[1]/2,dim[1]/2)
set_color(255,0,0)
fill_rect(-dim[0]/2,-dim[1]/2,318,212)
set_color(0,0,0)
fill_circle(0,-10,60)
fill_circle(-60,56,42)
fill_circle(60,56,42)
```




```
set_color(255,255,255)
fill_arc(-48,-76,96,66,0,360)
fill_circle(42,-34,18)
fill_circle(-42,-34,18)
fill_arc(-42,-28,48,66,0,360)
fill_arc(-6,-28,48,66,0,360)
```

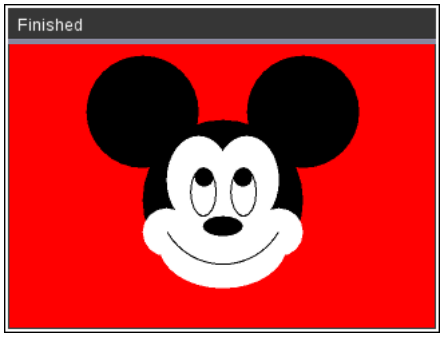



```
set_color(0,0,0)
draw_arc(-24,-22,19,36,0,360)
draw_arc(6,-22,19,36,0,360)
fill_arc(-15,-37,30,15,0,360)
draw_arc(-48,-58,96,96,210,120)
```



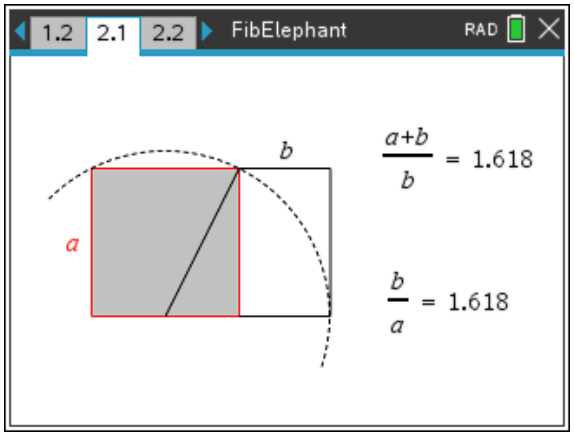
En tenslotte de pupillen van de ogen:

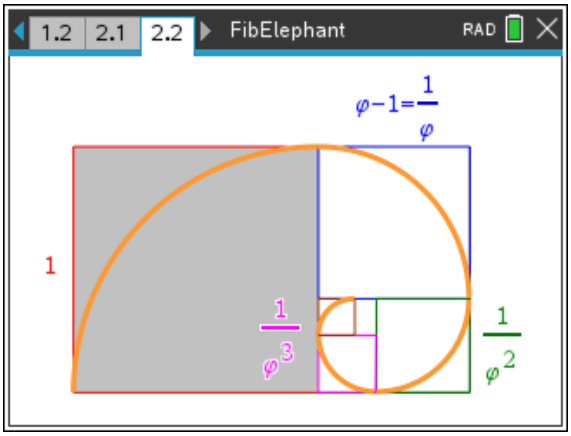
```
fill_circle(15,7.3,7)
fill_circle(-14,7.3, 7)
```



2.3. De olifant van Fibonacci

Een gouden rechthoek is een rechthoek waarvan de lengte en breedte zich verhouden als de gulden snede. $\varphi = \frac{1+\sqrt{2}}{5}$. Een gouden spiraal is een logaritmische spiraal die groeit met een factor φ.

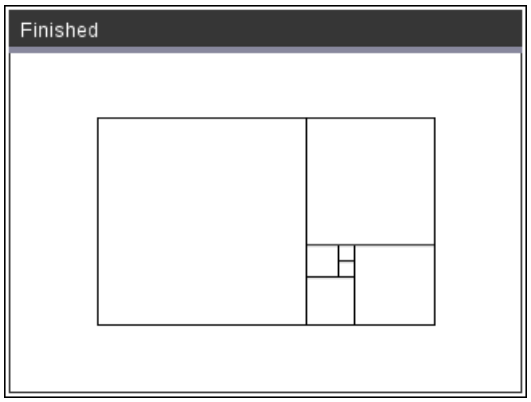


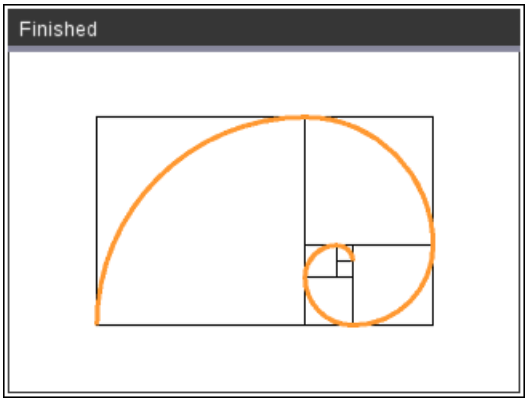


Vermits de rij met verhoudingen van opeenvolgende Fibonacci-getallen convergeert naar φ, kunnen we met de rij van Fibonacci een goudenspiraal benaderen.

De volgende code tekent de Fibonacci-spiraal:

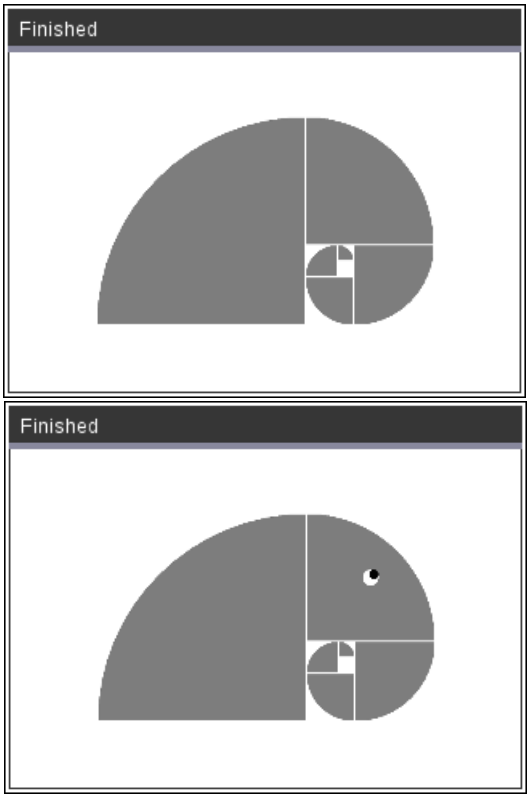
```
from ti_draw import *
dim=get_screen_dim()
set_window(-54,dim[0]-54,-42,dim[1]-42)
draw_rect(0,0,130,130)
draw_rect(130,50,80,80)
draw_rect(160,0,50,50)
draw_rect(130,0,30,30)
draw_rect(130,30,20,20)
draw_rect(150,40,10,10)
set_pen("medium","solid")
set_color(255,153,50)
z=130 ; draw_arc(0,0-z,2*z,2*z,90,90)
z=80 ; draw_arc(130-z,50-z,2*z,2*z,0,90)
z=50 ; draw_arc(160-z,0,2*z,2*z,270,90)
z=30 ; draw_arc(130,0,2*z,2*z,180,90)
z=20 ; draw_arc(130,30-z,2*z,2*z,90,90)
z=10 ; draw_arc(150-z,40-z,2*z,2*z,0,90)
```





Het inkleuren van de cirkelsectoren en wat aanpassen van de kleuren en de volgorde van plotten, geeft de volgende figuur – de olifant van Fibonacci.

```
from ti_draw import *

dim=get_screen_dim()
set_window(-54,dim[0]-54,-42,dim[1]-42)

set_color(125,125,125)
z=130 ; fill_arc(0,0-z,2*z,2*z,90,90)
z=80 ; fill_arc(130-z,50-z,2*z,2*z,0,90)
z=50 ; fill_arc(160-z,0,2*z,2*z,270,90)
z=30 ; fill_arc(130,0,2*z,2*z,180,90)
z=20 ; fill_arc(130,30-z,2*z,2*z,90,90)
z=10 ; fill_arc(150-z,40-z,2*z,2*z,0,90)

set_color(255,255,255)
draw_rect(0,0,130,130)
draw_rect(130,50,80,80)
draw_rect(160,0,50,50)
draw_rect(130,0,30,30)
draw_rect(130,30,20,20)
draw_rect(150,40,10,10)

fill_circle(170,90,5)
set_color(0,0,0)
fill_circle(172,92,3)
```



3. Interatieve grafsche algoritmes

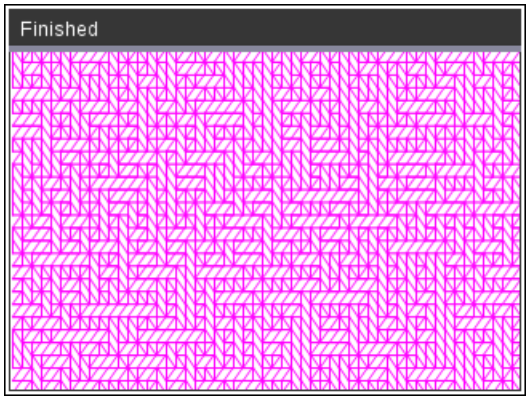
Voor de volgende voorbeelden gebruiken we de standaard vensterinstellingen.

3.1. At random creatief met tekst

Door de code at random te laten kiezen tussen twee karakters vullen we het grafische venster:

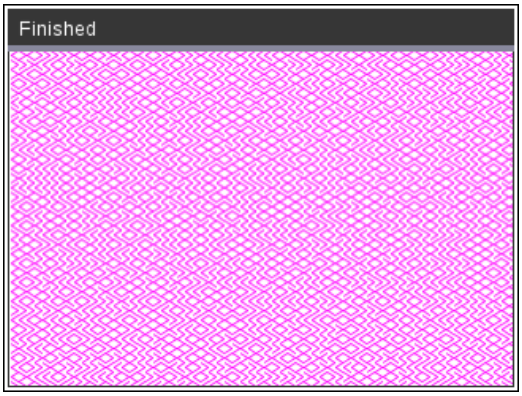
"N" of "Z"

```
from random import *
from ti_draw import *
use_buffer()
set_color(255,0,255)
for j in range(10,220,8):
    for i in range(0,318,6):
        if randint(0,1) > 0:
            draw_text(i,j,"N")
        else:
            draw_text(i,j,"Z")
    paint_buffer()
```



"<" of ">"

```
from random import *
from ti_draw import *
use_buffer()
set_color(255,0,255)
for j in range(10,220,6):
    for i in range(0,318,5):
        if randint(0,1) > 0:
            draw_text(i,j,"N")
        else:
            draw_text(i,j,"Z")
    paint_buffer()
```

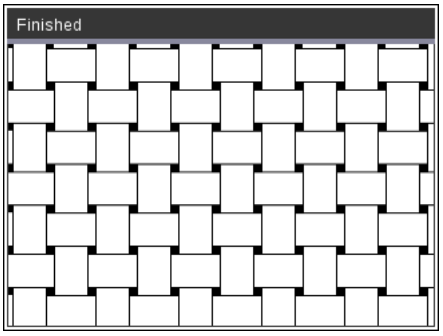


3.2. Modulo-matje weven

Met de volgende code tekenen we deze figuur:

Hiervoor verdelen we het scherm horizontaal en verticaal in d (=31) delen met s (=3) spatie tussen de verticale en horizontale banden.

```
from ti_draw import *
use_buffer()
d=31
s=3
```



Hoe coderen we de verticale banden?

De volgende lus geeft:

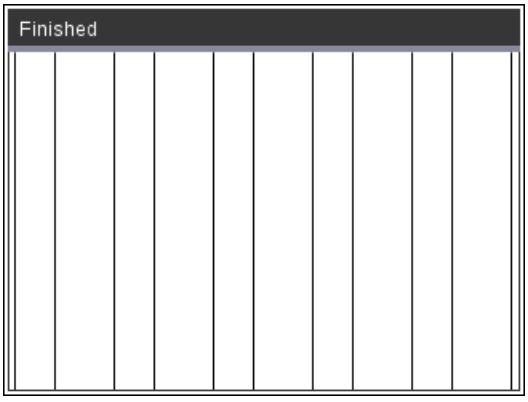
```
for i in range(0,318,d):
    x=i+s
    y=0
    draw_line(x,y,x,y+d)
paint_buffer()
```

We willen nu de x-coordinaat afwisselend vermeerderen met s en verminderen met s.
Dit kan als volgt m.b.v. modulo 2:

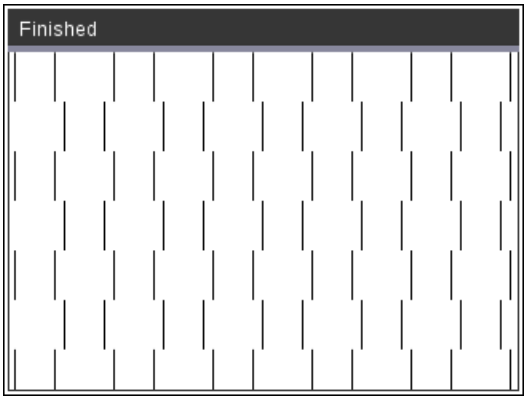
```
for i in range(0,318,d):
    if i%2 == 0:
        x=i+s
    else:
        x=i-s
    y=0
    draw_line(x,y,x,y+d)
paint_buffer()
```

Indien we dit verticaal herhalen met stap grootte d=31 geeft dit het linkse resultaat, niet wat we nodig hebben hier.
Daarom voegen we de variabele j toe (rechts) aan de modulo-check:

```
for j in range(0,212,d):
    for i in range(0,318,d):
        if i%2 == 0:
            x=i+s
        else:
            x=i-s
        y=j
        draw_line(x,y,x,y+d)
paint_buffer()
```

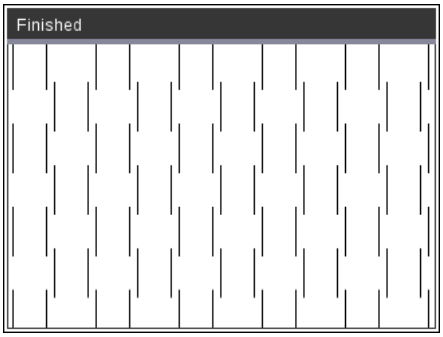



```
for j in range(0,212,d):
    for i in range(0,318,d):
        if (i+j)%2 == 0:
            x=i+s
        else:
            x=i-s
        y=j
        draw_line(x,y,x,y+d)
paint_buffer()
```



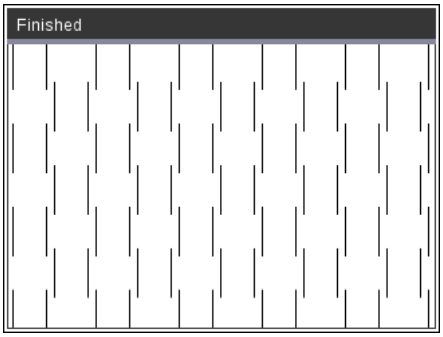
Als laatste stap verlengen we de lengte van de lijnstukkenmet 2*s, een afstand s omhoog en omlaag:

```
for j in range(0,212,d):
    for i in range(0,318,d):
        if (i+j)%2 == 0:
            x=i+s
        else:
            x=i-s
        y=j
        draw_line(x,y-s,x,y+d+s)
paint_buffer()
```

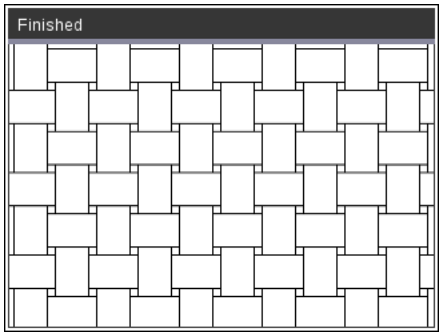


Voor de horizontale lijnstukken wisselen we de rollen van x en y om:

```
for j in range(0,212,d):
    for i in range(0,318,d):
        if (i+j)%2 == 0:
            y=j-s
        else:
            x=j+s
        x=i
        draw_line(x-s,y,x+ds,y)
paint_buffer()
```

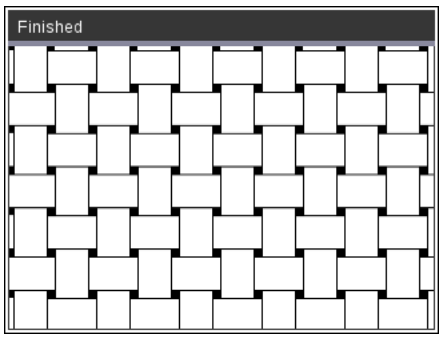


Beide blokken code voor de verticale en horizontale lijnstukken gecombineerd, geeft:



Rest nog het inkleuren van de vierkantjes met zijde 2*s:

```
for j in range(0,212,d):
    for i in range(0,318,d):
        fill_rect(i-s,j-s,2*s,2*s)
paint_buffer()
```



3.3. Verborgen cirkels

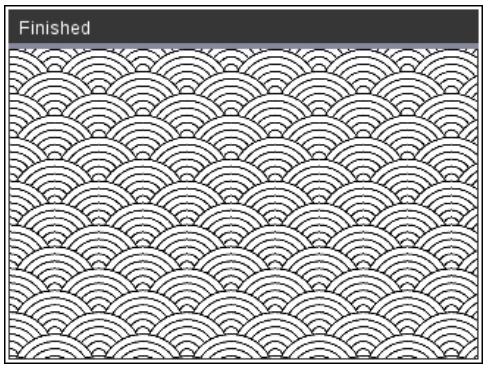
In het volgende voorbeeld wordt de kern van de herhalingen bepaald door zes cirkel met hetzelfde middelpunt en achtereenvolgend de stralen r = 30, r = 25, r = 20, r = 15, r = 10 en r = 5.

De x-coördinaten van de middelpunten laten we om de rij variëren tussen:

- even rij: x = 0, x = 60, x = 120, x = 180, x = 240, x = 330
- oneven rij: x = 30, x = 90, x = 150, x = 210, x = 270, x = 330

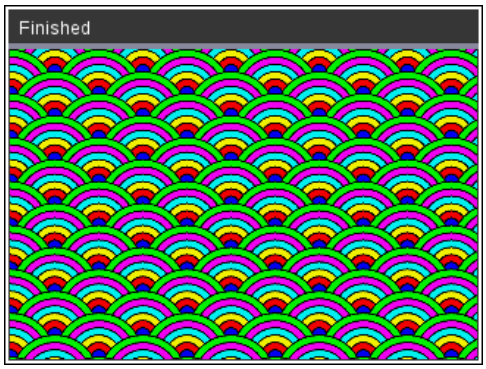
Voor de y-coordinaten verdelen we de hoogte van het scherm in 15.

```
from ti_draw import *
use_buffer()
# 15 rijen met figuren
for j in range(0,212/15+2):
    x=30*(j%2)
#   6 figuren per rij, niet altijd volledig zichtbaar
    for i in range(0,318,60):
#       Tekenen van 6 cirkels
        for r in range(30,0,-5):
            set_color(255,255,255)
            fill_circle(i+x,j*15,r)
            set_color(0,0,0)
            draw_circle(i+x,j*15,r)
paint_buffer()
```



Door set_color(255,255,255) te vervangen door set_color(75+6*r,75+6*r,75+6*r) worden er grayscale-schakeringen toegevoegd. Met het volgende kleurenpalet kan wat kleur aan de figuur toegevoegd worden.

```
from ti_draw import *
use_buffer()
def kleur(k):
    pal=[0,0,1,0,1,1,0,0]
    n=k%6
    set_color(240*pal[n],240*pal[n+1],240*pal[n+2])
k=0
for j in range(0,212/15+2):
    x=30*(j%2)
    for i in range(0,318,60):
        for r in range(30,0,-5):
            k=k+1
            kleur(k)
            fill_circle(i+x,j*15,r)
            set_color(0,0,0)
            draw_circle(i+x,j*15,r)
paint_buffer()
```



3.4. Optische misleidingen

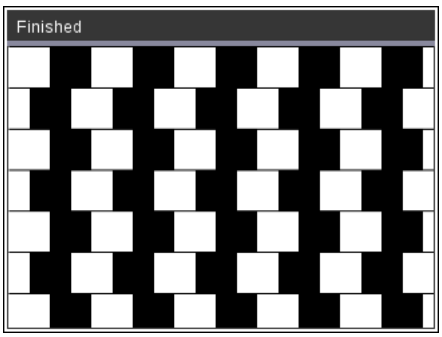
Voorbeeld 1
Parallelle vierkanten?

We coderen rijen van wit-zwarte vierkante (zijde 31):

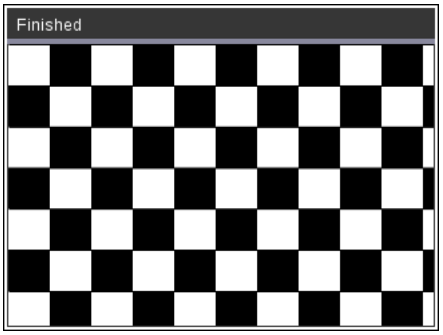
```
from ti_system import *.
from ti_draw import *

factor=float(input("Factor -1 ...  1: "))

use_buffer()

def vierkant(x,y,zijde):
♦♦px=[x,x+z,x+z,x,x]
♦♦py=[y,y,y+z,y+z,y]
♦♦fill_poly(px,py)

z=31

set_color(255,255,255)
fill_rect(0,0,318,212)

set_color(0,0,0)
for j in range(8):
♦♦draw_line(0,j*z,318,j*z)
♦♦x=z
♦♦if j%2 == 1:
♦♦♦♦x=-factor*z
♦♦for i in range(0,12,2):
♦♦♦♦vierkant(i*z+x,j*z,z)
paint_buffer()
```

Merk op dat het input-statement in de code moet staan voor de eerste functie van de TI Draw-module, daar zo'n functie het grafische venster activeert.

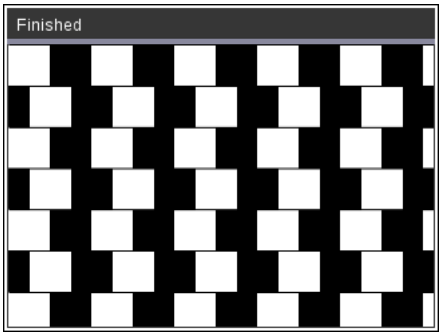
Factor = -0.5



Factor = 0



Factor = 0.5



Voorbeeld 2
Knipperende stippen?

De code voor het tekenen van het onderstaande raster bestaat uit de volgende delen:

Set up code en venster

```
from ti_draw import *
use_buffer()
zijde=40 ; b=3
fill_rect(0,0,318,212)
```

Horizontale lijnen (grijs)

```
set_color(160,160,160)
for j in range(0,212,zijde):
    fill_rect(0,j-b,318,2*b)
```

Verticale lijnen (grijs)

```
for i in range(0,318,zijde):
    fill_rect(i-b,0,2*b,212)
```

Stippen (wit)

```
set_color(240,240,240)
for j in range(0,212,zijde):
    for i in range(0,318,zijde):
        fill_circle(i,j,1.7*b)
paint_buffer()
```