

## Processing opstarten op cursuslaptop

Met een terminal kun je veel dingen doen, die soms niet met een normaal programma gedaan kunnen worden. Op de cursus start je Processing vanuit de terminal.

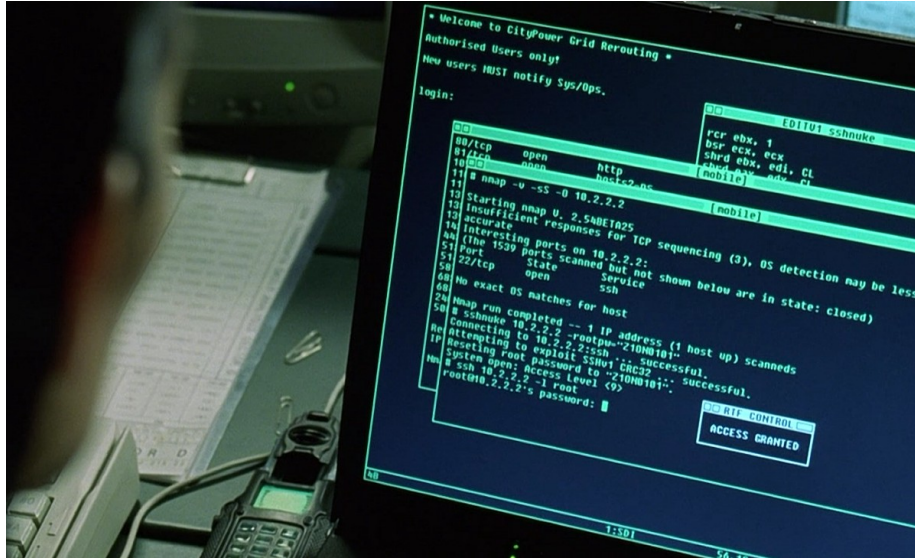


Figure 1: Hackers werken met een terminal

## Processing starten

Start een Terminal. Dit kan soms met Win+T, CTRL+ALT+T, of deze te vinden in de menuutjes, of te zoeken op het woord **Terminal**

Ga naar de folder waar Processing in staat. Hiervoor is het commando `cd`. De afkorting `cd` staat voor 'Change Directory'. "Change Directory" is Engels voor 'Verander van folder'.

Zo ga je naar de folder waar Processing instaat:

```
cd Programs/processing-3.1.1
```

Dit hoeft je niet zo te typen! Een terminal kan je woorden aanvullen als je op Tab drukt. Vaak is het volgende typen voldoende:

```
cd Progr[TAB]/pr[TAB]
```

Dit werkt ook als een andere versie van Processing op de computer staat :-)

Nu je in de juiste folder bent, start Processing:

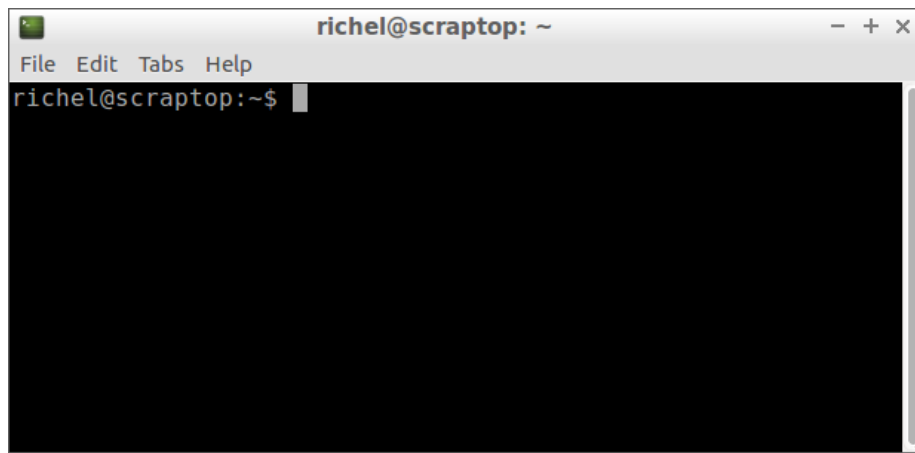


Figure 2: Een terminal

```
./processing
```

De ./ betekent 'Start hier'

Ook hier is Tab nuttig:

```
./p[TAB]
```

Je mag de terminal nu sluiten.

## Opdrachten

- Start Processing

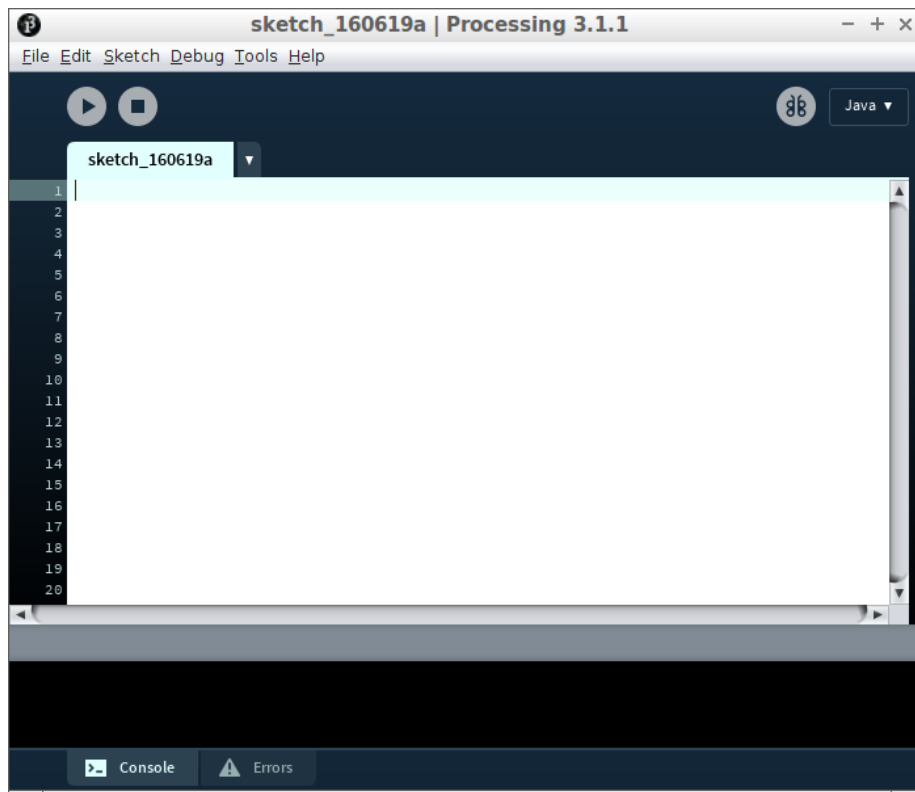


Figure 3: Processing zonder code

# Een Mooi Programma

Processing is een programmeertaal ontwikkeld voor kunstenaars en erg geschikt om games en mooie dingen mee te maken.

In deze les gaan we leren

- hoe we Processing opstarten
- hoe je code naar Processing kopieert
- hoe je het programma start

Zo ziet het programma eruit:

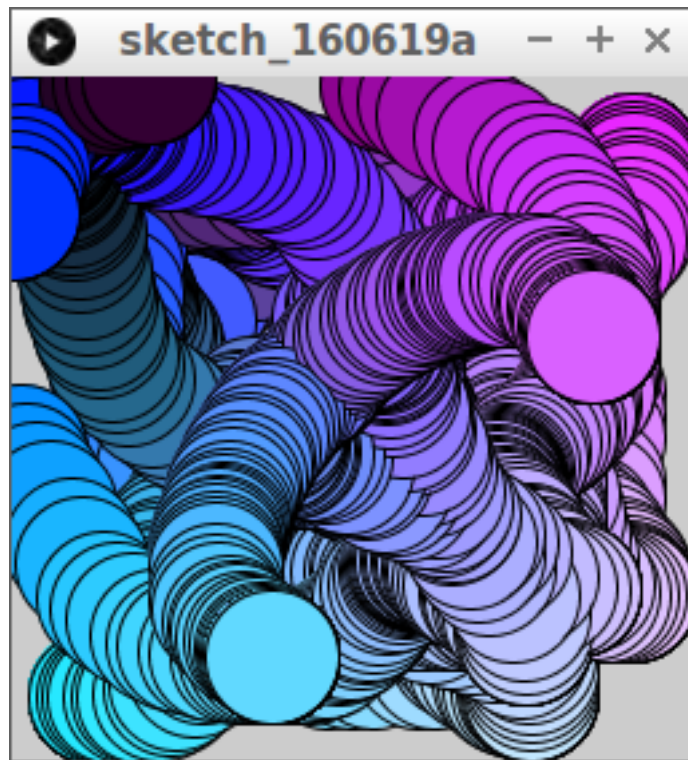


Figure 1: EenMooiProgramma

## Wat je nodig hebt

Je moet Processing op kunnen starten. Hoe dat moet, hangt af van het besturingssysteem:

- Processing opstarten op cursus laptop

- Processing installeren op eigen laptop met GNU/Linux
- Processing installeren op eigen laptop met Windows

## Code kopiëren

Processing begint met een leeg programma zonder code:

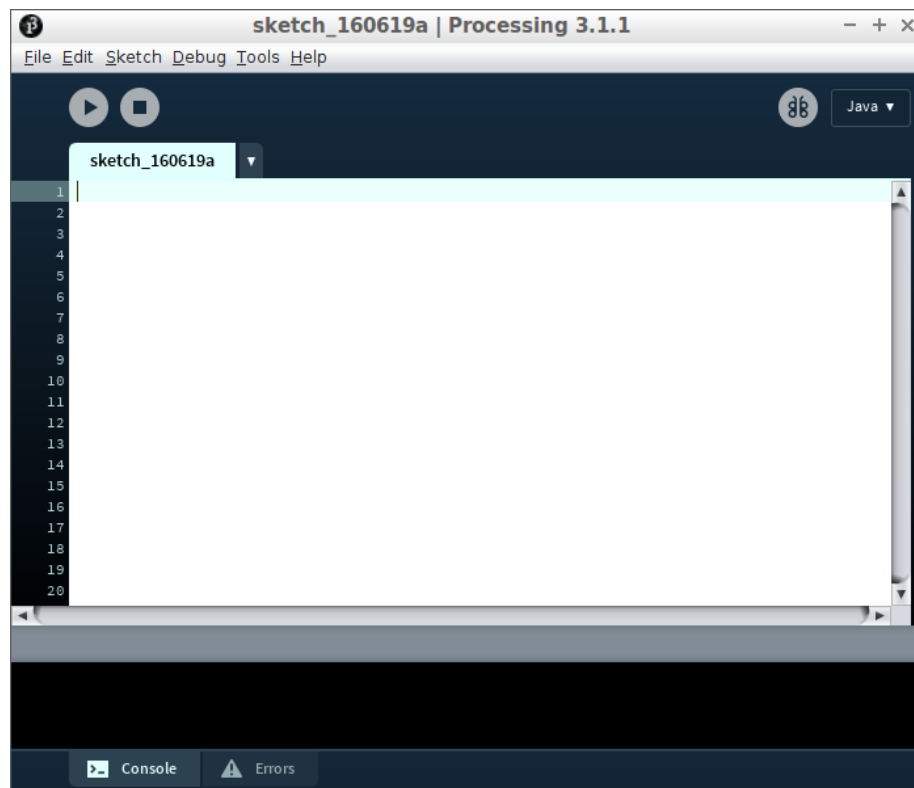


Figure 2: Processing zonder code

Dit is de programmeercode die we gaan gebruiken:

```
void setup()
{
  size(256,256);
}

void draw()
{
  fill(mouseX, mouseY, mouseX + mouseY);
  ellipse(mouseX, mouseY, 50, 50);
}
```

```

    fill(mouseY, mouseX, 255);
    ellipse(mouseY, mouseX, 50, 50);
}

```

Wat de code precies doet, leggen we later uit. Voor nu is het genoeg te weten dat het iets moois doet.

Om code te kopiëren gebruik je sneltoetsen:

- SHIFT + pijltjes: selecteren
- CTRL + A: alles selecteren
- CTRL + C: kopiëren van selectie
- CTRL + X: knippen van selectie
- CTRL + V: plakken van selectie
- Start Processing
- Kopieer deze code naar Processing

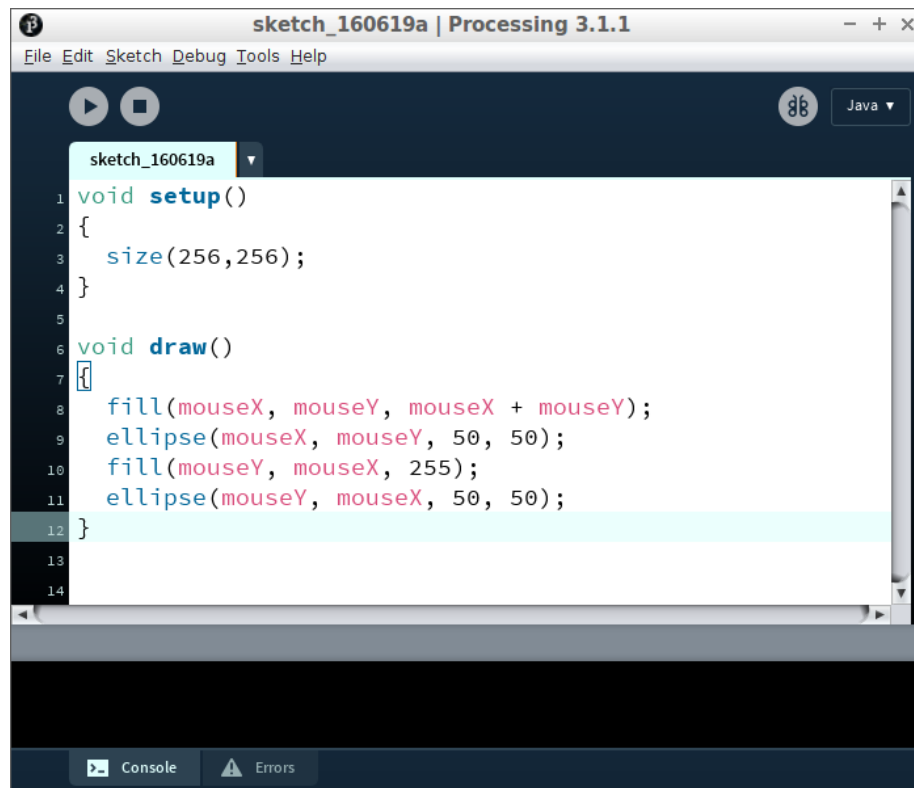


Figure 3: Processing met code

## Programma uitvoeren

- Klik op de 'Run' knop

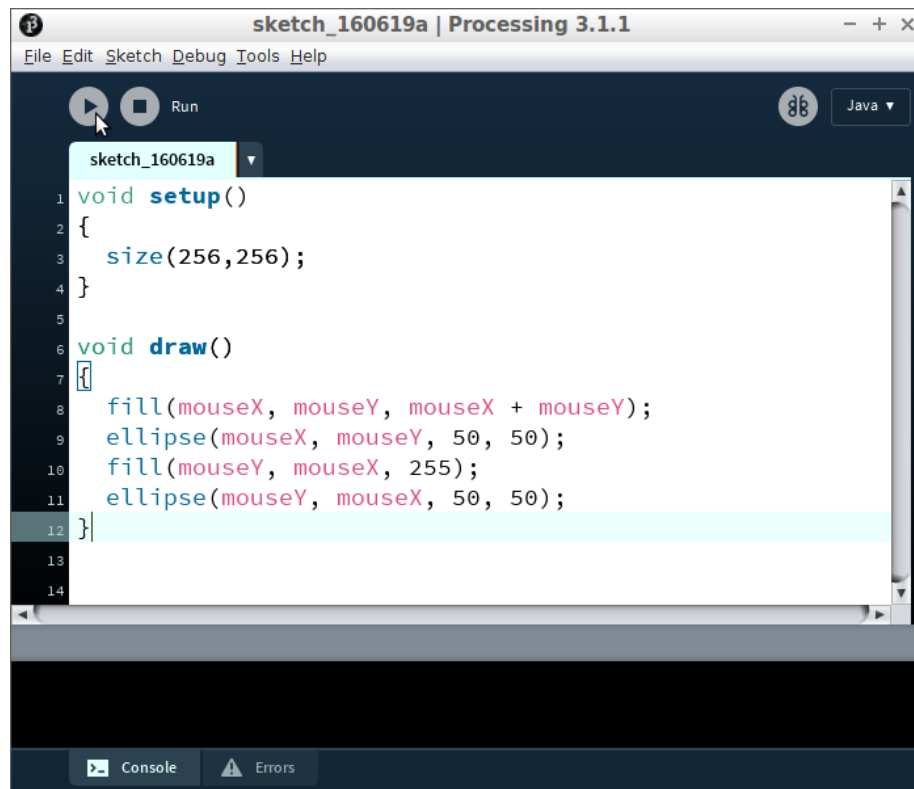


Figure 4: De Run knop

Als het goed is, zie je nu het programma!

## Sneltoetsen oefenen

- Werk met iemand samen. Hussel de code van de andere door de war, door deze te knippen/kopieren en te plakken. Repareer dan de code op je eigen computer

## Point

Processing is een programmeertaal ontwikkeld voor kunstenaars en erg geschikt om games en mooie dingen mee te maken.

In deze les gaan we leren

- wat pixels zijn
- hoe de pixels op een beeldscherm zitten
- hoe je puntjes tekent

Zo gaat het eruit zien:

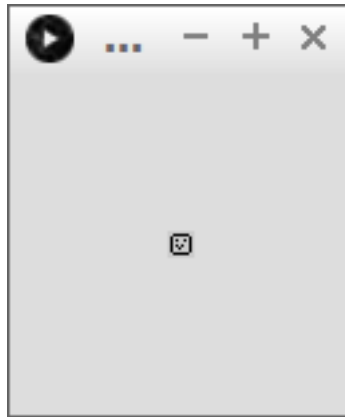


Figure 1: Point

## Pixels

Pixels zijn de vierkantjes waaruit je beeldscherm is opgebouwd. Je schermresolutie is het aantal pixels dat je scherm breed en hoog is. Hoe meer pixels je scherm heeft, hoe scherper je beeldscherm eruit ziet

Vroeger hadden computers een lage resolutie. Daarom zien oude spellen er wat blokkiger uit:

## Vragen

- Wat is de resolutie van jouw beeldscherm?
- Als een beeldscherm een resolutie heeft van 320 bij 200 pixels, hoeveel pixels heeft deze dan?



## Hoe zitten pixels op je beeldscherm

Elke pixel op je beeldscherm heeft een soort postcode. Deze postcode noemen we een coördinaat. Een coördinaat bestaat uit twee getallen. Dit zijn de coördinaten van alle pixels van een beeldscherm van vijf bij vijf pixels:

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)

Figure 2: Pixel coördinaat

Je ziet dat de laagste coördinaat  $(0,0)$  (zeg: ‘nul komma nul’) is.  $(0,0)$  zit in de linkerbovenhoek van je beeldscherm. Dan zie je dat het eerste getal is hoeveel pixels naar recht hiervan is. Zo zit  $(1,0)$  rechts van  $(0,0)$ . Het tweede getal is hoeveel pixels je onder  $(0,0)$  zit. Zo zit  $(0,1)$  onder  $(0,0)$ .

Op deze manier kun je elke pixel op je beeldscherm vinden.

## Vragen

- Welke coördinaten heb je nodig om een horizontale lijn te tekenen, die door  $(0,0)$  gaat en drie pixels lang is?

- Welke coördinaten heb je nodig om een verticale lijn te tekenen, die door (1,0) gaat en drie pixels hoog is?
- Welke coördinaten heb je nodig om een diagonale lijn te tekenen, die door (0,0) gaat en drie pixels lang is?
- Welke coördinaten heb je nodig om een vierkantje te tekenen, die door (0,0) gaat, twee pixels breed en twee pixels breed is?

## Puntjes tekenen

Dit is de programmeercode die je nodig hebt:

```
void setup()
{
  size(10,10);
}

void draw()
{
  // 0123456789
  // 0 .....
  // 1 ..XXXXXX..
  // 2 .X.....X.
  // 3 .X.X..X.X.
  // 4 .X.....X.
  // 5 .X.X.X..X.
  // 6 .X..X...X.
  // 7 .X.....X.
  // 8 ..XXXXXX..
  // 9 .....

  // Bovenkant hoofd
  point(2,1);
  point(3,1);
  point(4,1);
  point(5,1);
  point(6,1);
  point(7,1);

  // Rechterkant hoofd
  point(8,2);
  point(8,3);
  point(8,4);
  point(8,5);
  point(8,6);
```

```

    point(8,7);

    // Onderkant hoofd
    point(2,8);
    point(3,8);
    point(4,8);
    point(5,8);
    point(6,8);
    point(7,8);

    // Linkerkant hoofd
    point(1,2);
    point(1,3);
    point(1,4);
    point(1,5);
    point(1,6);
    point(1,7);

    // Ogen
    point(3,3);
    point(6,3);

    // Mond
    point(3,5);
    point(4,6);
    point(5,5);

}

```

Dit is wat de code doet:

- `void setup() {}`: de setup functie, de computer voert een keer alles tussen de accolades uit
- `size(10, 10)`: maak een scherm van 10 pixels breed en 10 pixels hoog
- `;`: de puntkomma geeft in Processing het einde aan van een zin. Dit is ongeveer hetzelfde met een punt in schrijftaal.
- `void draw() {}`: de draw functie, de computer voert steeds alles tussen de accolades uit
- `//`: commentaar: de rest van de regel wordt niet gelezen door de computer. Hier kun je dingen neerzetten die speciaal bedoelt zijn voor mensen, zoals jezelf. In dit programma zie je dat het poppetje is getekent in commentaar, met de coördinaten aan de zijkant. Ook zie je dat de programmeur zegt wat er getekend wordt: zo kun je de fouten sneller vinden
- `point(2,1)`: teken een puntje op coördinaat (2,1).

## Vragen

- Start Processing. Kopieer deze code en run het programma
- Laat de smiley een pixel breder lachen
- Draai de lach verticaal om, zodat de smiley sip gaat kijken
- Geef de smiley punk haar door drie pixels bovenop het hoofd erbij te tekenen
- Geef de smiley een sik van een pixel

## Line

Zonder lijnen kun je bijna geen games maken. Een van de allereerste successgames, Asteroids, bestond vooral uit lijnen:

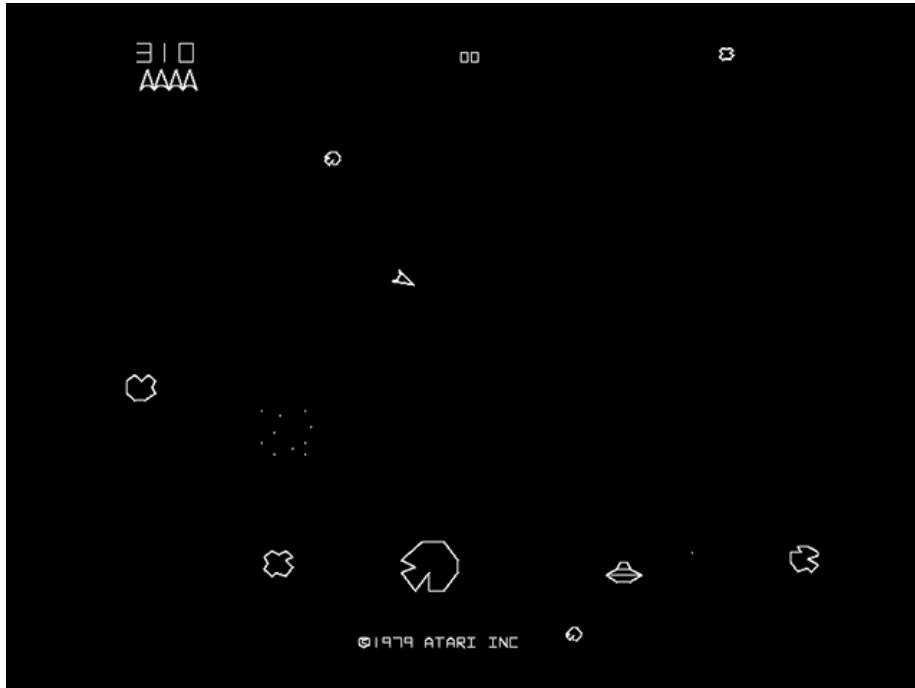


Figure 1: Asteroids

Je kunt een lijn tekenen met een boel puntjes, maar de `line` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je lijnen tekent

Zo gaat het eruit zien:

Kun je nog geen puntjes tekenen? Ga dan naar de les waarin je puntjes leert tekenen

## Lijnen

Een lijn bestaat uit pixels. Om een lijn te tekenen, moet je een beginpixel en eindpixel kiezen. Processing tekent dan zelf de pixels ertussenin.

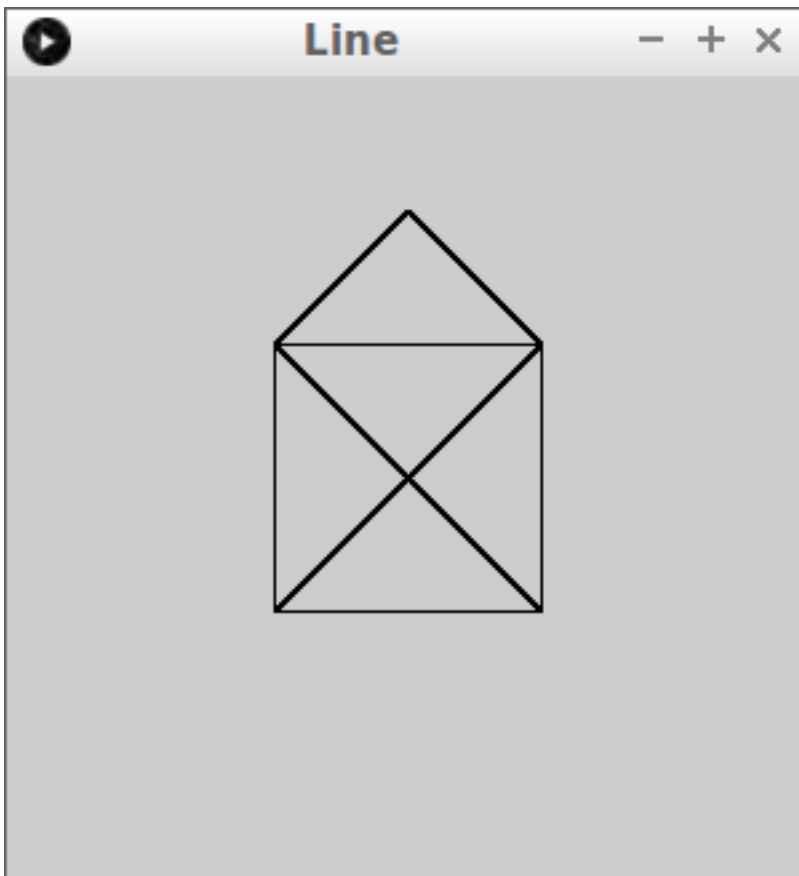


Figure 2: Line

Om in Processing een lijn te tekenen, gebruik je de functie `line`. De functie `line` heeft vier getallen nodig. Deze vier getallen zijn twee coördinaten achter elkaar. De eerste twee getallen zijn de coördinaat van het ene eind van de lijn. De laatste twee getallen zijn de coördinaat van het ander eind van de lijn.

Hier zie je een lijn die gaat van coördinaat (1,2) naar (3,2):

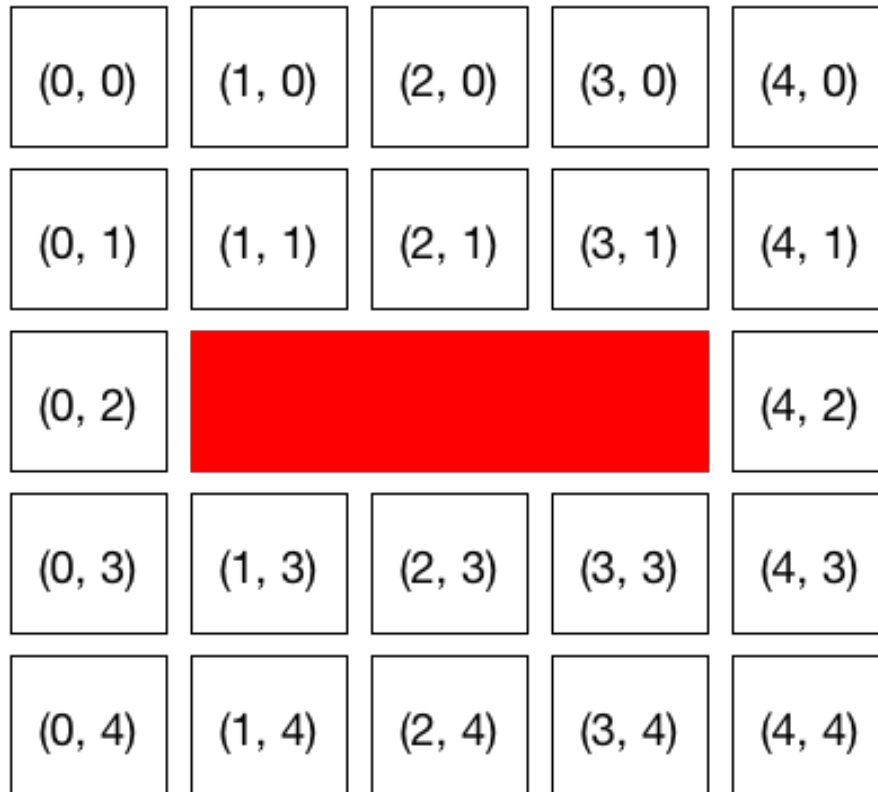


Figure 3: Lijn 1

In Processing teken je deze lijn met:

```
line(1,2,3,2);
```

De volgorde van de twee coördinaten maakt niet uit.

Lijnen kunnen ook schuin gaan.

Hier zie je een lijn die gaat van coördinaat (2,4) naar (1,1):

In Processing teken je deze lijn met:

```
line(2,4,1,1);
```

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)		(2, 1)	(3, 1)	(4, 1)
(0, 2)		(2, 2)	(3, 2)	(4, 2)
(0, 3)	(1, 3)		(3, 3)	(4, 3)
(0, 4)	(1, 4)		(3, 4)	(4, 4)

Figure 4: Lijn 2



## Vragen

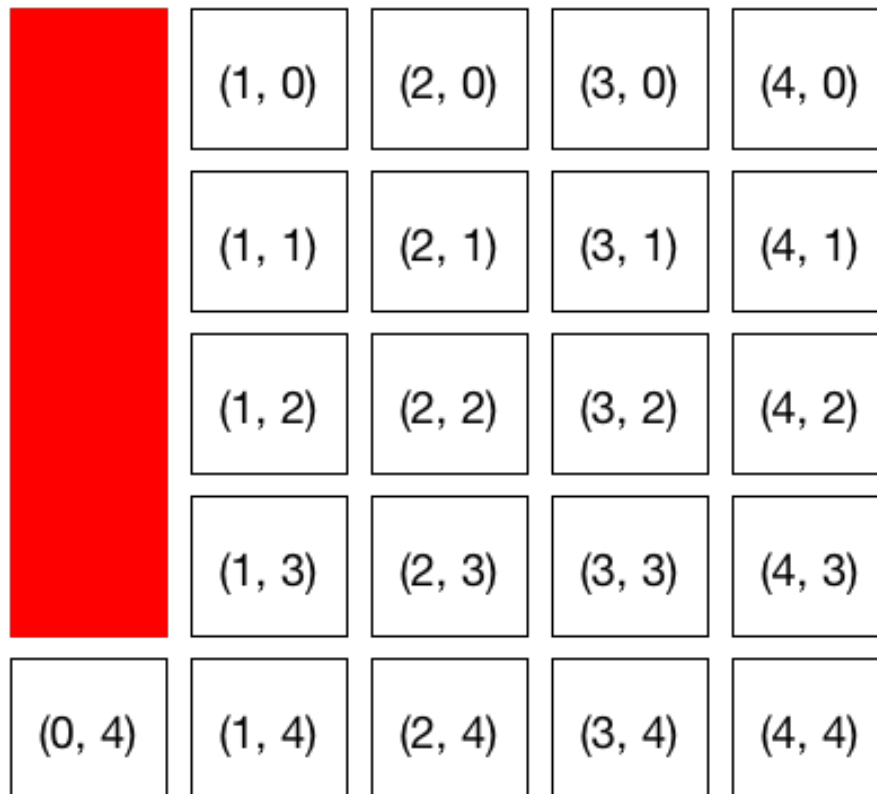


Figure 5: Lijn 3

- 1. Hierboven staat een lijn. Wat zijn de begin- en eindcoördinaat van die lijn?
- 2. Hierboven staat een lijn. Wat zijn de begin- en eindcoördinaat van die lijn?
- 3. Hierboven staat een lijn. Wat zijn de begin- en eindcoördinaat van die lijn?
- 4. Een lijn gaat van coördinaat (0,0) naar (10,0). In welke richting gaat de lijn? Wat is het Processing commando?
- 5. Een lijn gaat van coördinaat (0,0) naar (0,10). In welke richting gaat de lijn? Wat is het Processing commando?
- 6. Een lijn gaat van coördinaat (0,0) naar (10,10). In welke richting gaat de lijn? Wat is het Processing commando?

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)	(1, 1)	(2, 1)		(4, 1)
(0, 2)	(1, 2)	(2, 2)		(4, 2)
(0, 3)	(1, 3)	(2, 3)		(4, 3)
(0, 4)	(1, 4)	(2, 4)		(4, 4)

Figure 6: Lijn 4

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)	(1, 1)		(3, 1)	(4, 1)
(0, 2)		(2, 2)	(3, 2)	(4, 2)
	(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)

Figure 7: Lijn 5

- 7. Een lijn gaat van coördinaat (30,20) naar (20,20). In welke richting gaat de lijn? Wat is het Processing commando?
- 8. Een lijn gaat van coördinaat (10,20) 20 pixels naar rechts. Welke coördinaat heeft het eindpunt? Wat is het Processing commando?
- 9. Een lijn gaat van coördinaat (10,30) 10 pixels naar rechtsomhoog. Welke coördinaat heeft het eindpunt? Wat is het Processing commando?

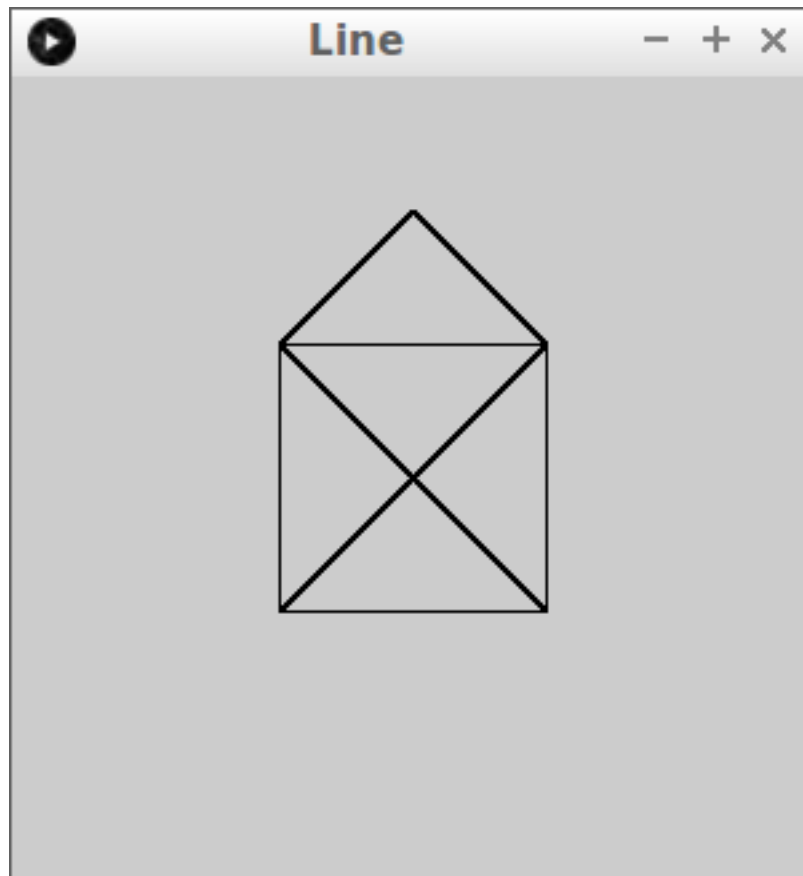


Figure 8: Line

- 10. Hierboven staat een tekening. Maak deze tekening na in Processing

### Oplossing

- 1. (0,0) en (0,3)

- 2. (3,1) en (3,4)
- 3. (2,1) en (0,3)
- 4. Van links naar rechts/horizontaal. `line(0,0,10,0)`
- 5. Van onder naar boven/verticaal. `line(0,0,0,10)`
- 6. Schuin/diagonaal. `line(0,0,0,10)`
- 7. Van rechts naar links/horizontaal. `line(30,20,20,20)`
- 8. (30,20). `line(10,20,30,20)`
- 9. (20,20). `line(10,30,20,20)`
- 10. Zie hieronder:

```
void setup()
{
  size(300,300);
}

void draw()
{
  //
  //   a
  //  / \
  // e---b
  //  | \ / |
  //  | X |
  //  | / \ |
  // d---c
  //
  // a: (150, 50)
  // b: (200,100)
  // c: (200,200)
  // d: (100,200)
  // e: (100,100)

  //Van a naar b
  line(150,50,200,100);
  //Van b naar c
  line(200,100,200,200);
  //Van c naar d
  line(200,200,100,200);
  //Van d naar e
  line(100,200,100,100);
  //Van e naar a
  line(100,100,150,50);
}
```

```
//Van b naar d  
line(200,100,100,200);  
//Van b naar e  
line(200,100,100,100);  
//Van c naar e  
line(200,200,100,100);  
}
```

## backGround

Zonder kleur zien games er minder mooi uit.

Een van de allereerste games met kleur heette Moria:



Figure 1: Moria

Om goed kleuren te kunnen gebruiken, moeten we leren hoe kleuren werken.

In deze les gaan we leren

- hoe kleuren werken

Zo gaat het eruit zien:

Je hoeft maar weinig te weten, behalve hoe je een mooi programma maakt

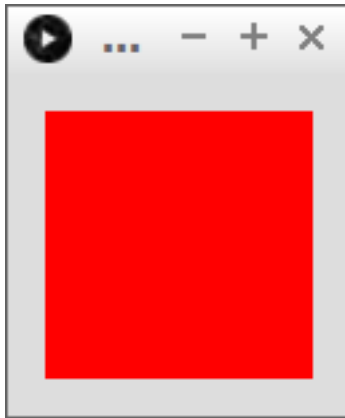


Figure 2: background

## Kleuren

Als je goed naar een beeldscherm kijkt, zie je dat elke pixel uit drie lampjes bestaat:

De lampjes hebben de kleuren rood, groen en blauw.

Omdat de lampjes zo klein zijn, ziet ons oog van afstand de menging van deze drie lampjes. Zo zien de lampjes rood en groen er samen uit als geel. Hier zie je hoe de kleuren mengen:

Om wit te krijgen, heb je alledrie de kleuren nodig.

## Vragen

- 1. Welke drie kleuren lampjes heeft een pixel?
- 2. Met welke lampjes samen maak je geel?
- 3. Met welke lampjes samen maak je cyaan/lichtblauw?
- 4. Met welke lampjes samen maak je magenta/paars?
- 5. Met welke lampjes samen maak je wit?
- 6. Met welke lampjes samen maak je zwart?
- 7. Met welke lampjes samen maak je grijs?
- 8. Met welke lampjes samen maak je oranje?



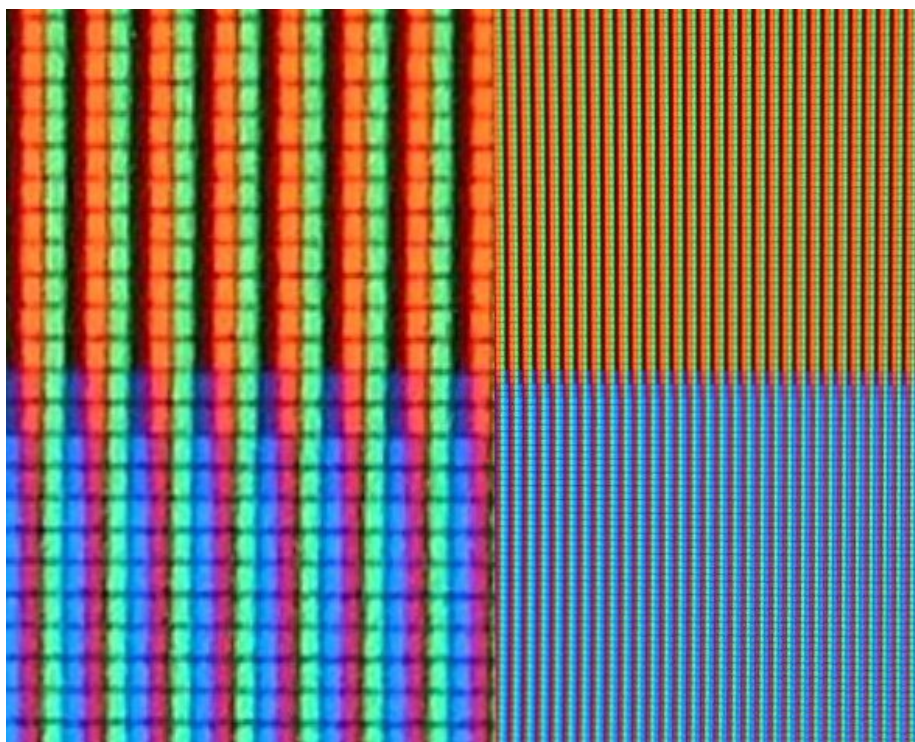


Figure 3: RGB pixels

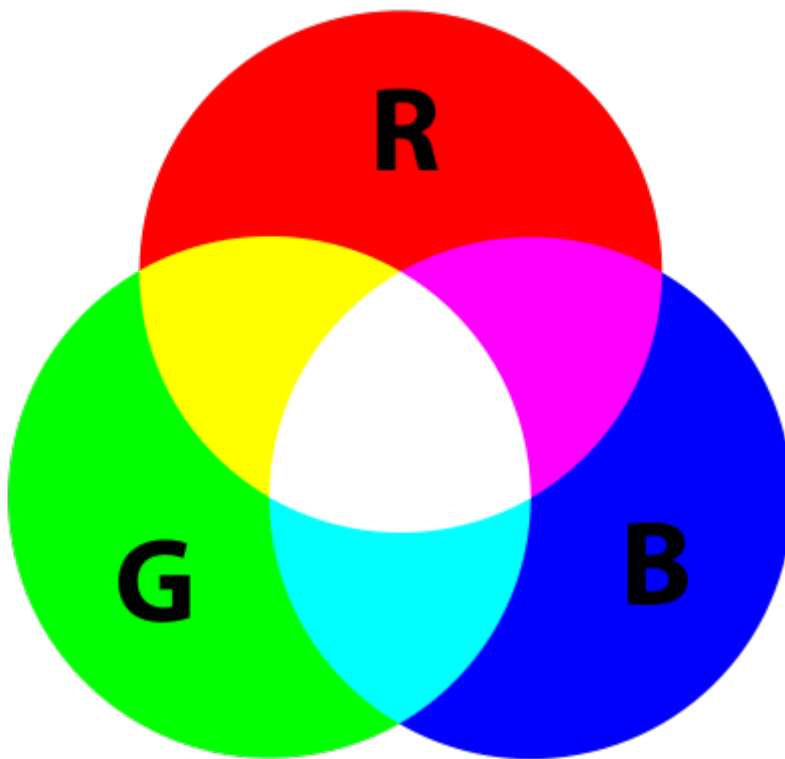


Figure 4: Additieve kleuren

## Antwoorden

- 1. Rood, groen en blauw
- 2. Rood en groen
- 3. Groen en blauw
- 4. Rood en blauw
- 5. Rood en groen en blauw
- 6. Met geen lampjes: als alle lampjes uit zijn, is het zwart
- 7. Met rood en groen en blauw, maar dan moeten ze niet op hun hardst branden
- 8. Met rood op z'n hardst en groen op halve kracht

## background

In Processing is er een functie om de achtergrond een kleur te geven. Deze functie heet **background**. **background** is Engels voor 'achtergrond'. **background** is een functie die drie getallen nodig heeft. Deze drie getallen bepalen hoe hard de rode, groene en blauwe lampjes gaan schijnen. Deze drie getallen noemen we de RGB waarde. Met het getal nul zeg je dat een lampje uit staat. Met het getal 255 zeg je dat een lampje aan staat. Met getallen tussen nul en 255 kun je het lampje ertussenin laten branden.

Met deze Processing code krijg je een rode achtergrond:

```
void setup()
{
  size(100,100);
}

void draw()
{
  background(255,0,0);
}
```

## Opdracht

- 1. Kopieer deze code in Processing en start de code
- 2. Wat is een RGB waarde?
- 3. Wat is de RGB waarde van groen? Maak in Processing een groene achtergrond

- 4. Wat is de RGB waarde van blauw? Maak in Processing een blauwe achtergrond
- 5. Wat is de RGB waarde van geel? Maak in Processing een gele achtergrond
- 6. Wat is de RGB waarde van cyaan/lichtblauw? Maak in Processing een cyane/lichtblauwe achtergrond
- 7. Wat is de RGB waarde van magenta/paars? Maak in Processing een magenta/paarse achtergrond
- 8. Wat is de RGB waarde van wit? Maak in Processing een witte achtergrond
- 9. Wat is de RGB waarde van zwart? Maak in Processing een zwart achtergrond
- 10. Wat is de RGB waarde van grijs? Maak in Processing een grijze achtergrond
- 11. Wat is de RGB waarde van donkerrood? Maak in Processing een donkerrode achtergrond
- 12. Wat is de RGB waarde van oranje? Maak in Processing een oranje achtergrond

## Oplossingen

- 1. OK
- 2. De Rood-Groen-Blauw waarde. Dit zijn drie getallen van nul tot en met 255 die bepalen hoe hard de drie kleurenlampjes branden
- 3. `background(0,255,0)`
- 4. `background(0,0,255)`
- 5. `background(255,255,0)`
- 6. `background(0,255,255)`
- 7. `background(255,255,0)`
- 8. `background(255,255,255)`
- 9. `background(0,0,0)`
- 10. `background(128,128,128)`, maar andere getallen tussen 0 en 255 zijn ook goed. Als ze maar alledrie gelijk zijn
- 11. `background(128,0,0)`, maar het eerste getal mag ook een ander getal tussen de 0 en 255 zijn

- 12. `background(255,128,0)`, maar het tweede getal mag ook een ander getal in de buurt van 128 zijn

## stroke

Zonder kleur zien games er minder mooi uit.

Een van de allereerste games met kleur heette Moria:



Figure 1: Moria

In deze les gaan we leren

- hoe je een lijn een kleur kan geven.

Zo gaat het eruit zien:

Weet je nog niet hoe kleuren werken, ga dan naar de les background

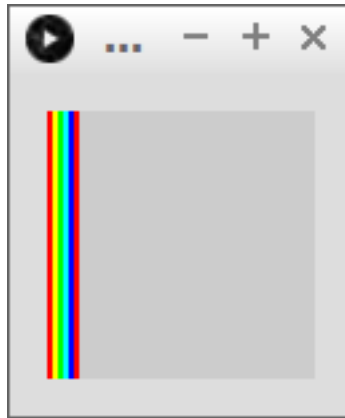


Figure 2: Stroke

## **stroke**

In Processing is er een functie om lijnen een kleur te geven. Deze functie heet **stroke**. **stroke** is Engels voor '(penseel)streek'. **stroke** is een functie die drie getallen nodig heeft. Deze drie getallen zijn de RGB waarden.

Met deze Processing code krijg je een rode lijn:

```
void setup()
{
  size(100,100);
}

void draw()
{
  stroke(255,0,0);
  line(10,20,30,40);
}
```

Met **stroke** zeg je: 'vanaf nu wil ik deze lijnkleur'. Hieronder zie je hoe je twee groene en een blauwe lijn tekent:

```
void setup()
{
  size(100,100);
}

void draw()
{
  stroke(0,255,0);
  line(10,20,30,40);
}
```

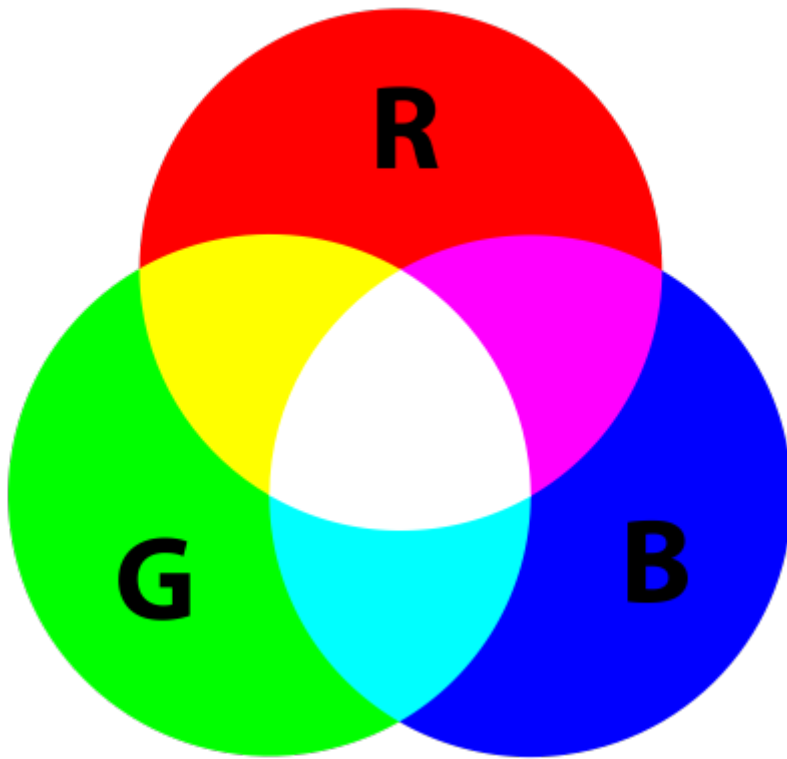


Figure 3: Kleurencirkel



```

    line(50,60,70,80);
    stroke(0,0,255);
    line(90,10,20,30);
}

```

## Opdracht

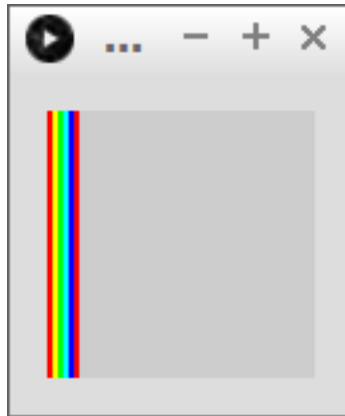


Figure 4: Stroke

- 1. Maak in Processing bovenstaande tekening na. Het venster is honderd bij honderd pixels. Elke kleur is met twee lijnen getekend. De kleuren zijn rood, geel, groen, cyaan, blauw, magenta

## Oplossing

```

void setup()
{
    size(100,100);
}

void draw()
{
    stroke(255,0,0);
    line(0,0,0,100);
    line(1,0,1,100);
    stroke(255,255,0);
    line(2,0,2,100);
    line(3,0,3,100);
    stroke(0,255,0);
    line(4,0,4,100);
}

```

```
line(5,0,5,100);  
stroke(0,255,255);  
line(6,0,6,100);  
line(7,0,7,100);  
stroke(0,0,255);  
line(8,0,8,100);  
line(9,0,9,100);  
stroke(255,0,0);  
line(10,0,10,100);  
line(11,0,11,100);  
}
```

## Rect

Vierkanten worden veel gebruikt in games.

Hier zie je een van de beroemdste games ooit:



Figure 1: Tetris

Je kunt een vierkant tekenen met vier lijnen, maar de `rect` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je lijnen tekent

Zo gaat het eruit zien:

Kun je nog geen lijnen tekenen? Ga dan naar de les waarin je lijnen leert tekenen

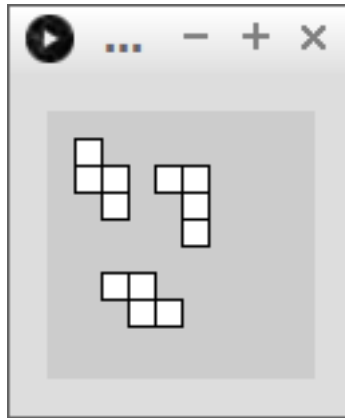


Figure 2: Rect

## Rechthoeken

Een rechthoek bestaat uit vier lijnen. Om een rechthoek te tekenen, moet je een coördinaat, breedte en hoogte geven.

Om in Processing een rechthoek te tekenen, gebruik je de functie `rect`. De functie `rect` heeft vier getallen nodig. De eerste twee getallen zijn de coördinaat van de linkerbovenhoek van de rechthoek. Het derde getal is de breedte van de rechthoek. Het vierde getal is de hoogte van de rechthoek.

Hier zie je een rechthoek met coördinaat (1,2), een breedte van drie pixels en hoogte van vier pixels:

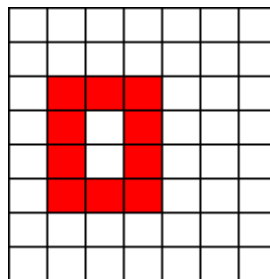


Figure 3: Rechthoek 1

In Processing teken je deze rechthoek met:

```
rect(1,2,3,4);
```

Hier is nog een rechthoek:

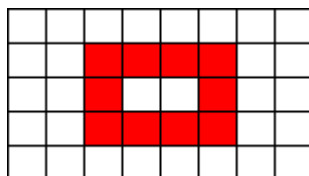


Figure 4: Rechthoek 2

De linkerbovenhoek heeft coördinaat (2,1), hij is vier pixels breed en drie pixels hoog.

## Vragen

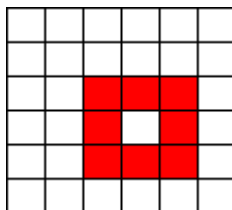


Figure 5: Rechthoek 3

- 1. Hierboven staat een rechthoek. Wat is de coördinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?

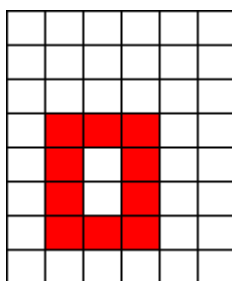


Figure 6: Rechthoek 4

- 2. Hierboven staat een rechthoek. Wat is de coördinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?
- 3. Hierboven staat een rechthoek. Wat is de coördinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?
- 4. Een rechthoek heeft als coördinaat (0,0), is twee pixels breed en drie pixels hoog. Wat is het Processing commando?

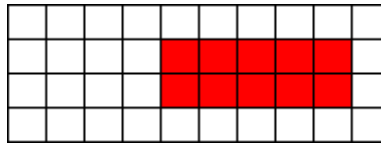


Figure 7: Rechthoek 5

- 5. Een rechthoek heeft als coördinaat (1,2), is drie pixels breed en vier pixels hoog. Wat is het Processing commando?
- 6. Een rechthoek heeft als coördinaat (10,20), is dertig pixels breed en veertig pixels hoog. Wat is het Processing commando?

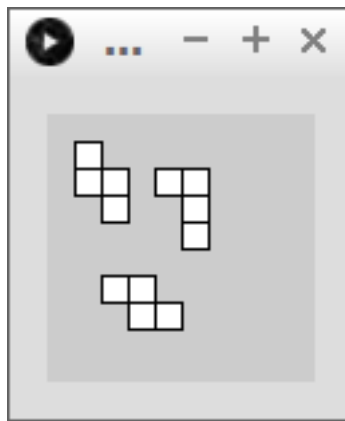


Figure 8: Rect

- 10. Hierboven staat een tekening. Maak deze tekening na in Processing

## Oplossing

- 10. Zie hieronder:

```
void setup()
{
  size(100,100);
}

void draw()
{
  rect(10,10,10,10);
  rect(10,20,10,10);
  rect(20,20,10,10);
}
```

```
rect(20,30,10,10);

rect(40,20,10,10);
rect(50,20,10,10);
rect(50,30,10,10);
rect(50,40,10,10);

rect(20,60,10,10);
rect(30,60,10,10);
rect(30,70,10,10);
rect(40,70,10,10);
}
```

## Ellipse

Cirkels en ovalen worden veel gebruikt in games.

Hier zie je een beroemde game, Bubble Bobble, dat veel met cirkels werkt:



Figure 1: Bubble Bobble

Je kunt een ovaal tekenen met heel veel puntjes, maar de `ellipse` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je ovalen tekent

Zo gaat het eruit zien:

Kun je nog geen rechthoeken tekenen? Ga dan naar de les waarin je rechthoeken leert tekenen

## Ovalen

Een ovaal heeft een middelpunt, breedte en hoogte. Om een ovaal te tekenen, moet je een coördinaat, breedte en hoogte geven.

Om in Processing een ovaal te tekenen, gebruik je de functie `ellipse`. De functie `ellipse` heeft vier getallen nodig. De eerste twee getallen zijn de coördinaat



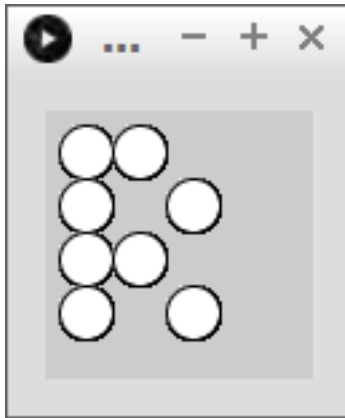


Figure 2: Ellipse

van het midden van de ovaal. Het derde getal is de breedte van de ovaal. Het vierde getal is de hoogte van de ovaal.

Hier zie je een ovaal met middelpunt (3,2), een breedte van vijf pixels en hoogte van drie pixels:

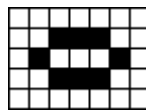


Figure 3: Ovaal 1

In Processing teken je deze ovaal met:

```
ellipse(3,2,5,3);
```

Hier is nog een ovaal:

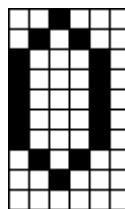


Figure 4: Ovaal 2

Het middelpunt heeft coördinaat (2,4), hij is vijf pixels breed en negen pixels hoog.

## Vragen

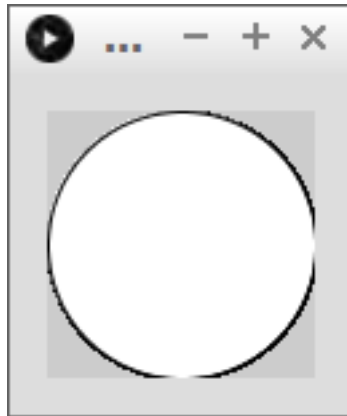


Figure 5: Ovaal 3

- 1. Je wilt bovenstaande plaatje namaken. Het venster is 100 pixels breed en 100 pixels hoog. Wat is het middelpunt van de cirkel? Hoe breed is de cirkel? En hoe hoog? Hoe maak je dit in Processing?



Figure 6: Ovaal 4

- 2. Je wilt bovenstaande plaatje namaken. Het venster is 200 pixels breed en 100 pixels hoog. Wat is het middelpunt van de cirkel? Hoe breed is de cirkel? En hoe hoog? Hoe maak je dit in Processing?
- 3. Je wilt bovenstaande plaatje namaken. Het venster is 200 pixels breed en 100 pixels hoog. Wat zijn de middelpunten van de cirkels? Hoe breed zijn de cirkels? En hoe hoog? Hoe maak je dit in Processing?
- 4. Hierboven staat een tekening. Maak deze tekening na in Processing



Figure 7: Ovaal 5

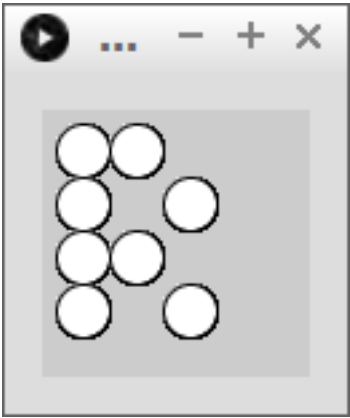


Figure 8: Ellipse

## Oplossing

- 1. Zie hieronder:

```
void setup() {  
  size(100, 100);  
}  
  
void draw() {  
  ellipse(50, 50, 100, 100);  
}
```

- 2. Zie hieronder:

```
void setup() {  
  size(200, 100);  
}  
  
void draw() {  
  ellipse(100, 50, 200, 100);  
}
```

- 3. Zie hieronder:

```
void setup() {  
  size(200, 100);  
}  
  
void draw() {  
  ellipse( 50, 50, 100, 100);  
  ellipse(150, 50, 100, 100);  
}
```

- 4. Zie hieronder:

```
void setup()  
{  
  size(100, 100);  
}  
  
void draw()  
{  
  ellipse(15,15, 20, 20);  
  ellipse(15,35, 20, 20);  
  ellipse(15,55, 20, 20);  
  ellipse(15,75, 20, 20);  
  
  ellipse(35, 15, 20, 20);  
  ellipse(55, 35, 20, 20);  
}
```

```
    ellipse(35, 55, 20, 20);  
    ellipse(55, 75, 20, 20);  
}
```

## fill

Een rechthoek of een ovaal kan ook ingekleurd worden.

Een van de beroemdste games ooit heeft bijvoorbeeld veel ingekleurde vierkanten:

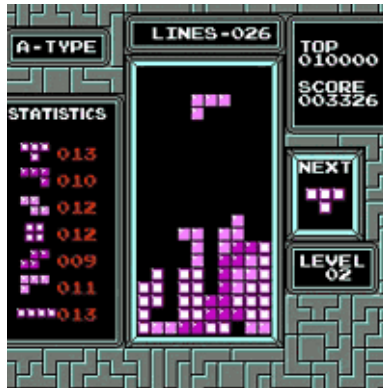


Figure 1: Tetris

In deze les gaan we leren

- hoe we de invulkleur van vierkanten en ovalen instellen

Zo gaat het eruit zien:

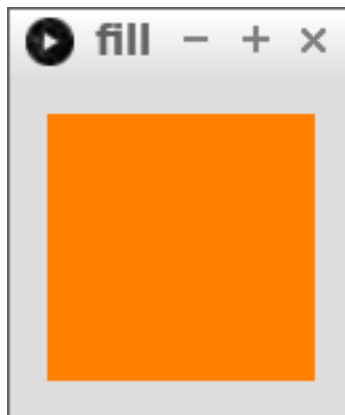


Figure 2: Fill

Weet je nog niet hoe je een lijn een kleur kan geven? Ga dan naar [stroke](#)

## Kleuren

De lampjes hebben de kleuren rood, groen en blauw.

Omdat de lampjes zo klein zijn, ziet ons oog van afstand de menging van deze drie lampjes. Zo zien de lampjes rood en groen er samen uit als geel. Hier zie je hoe de kleuren mengen:

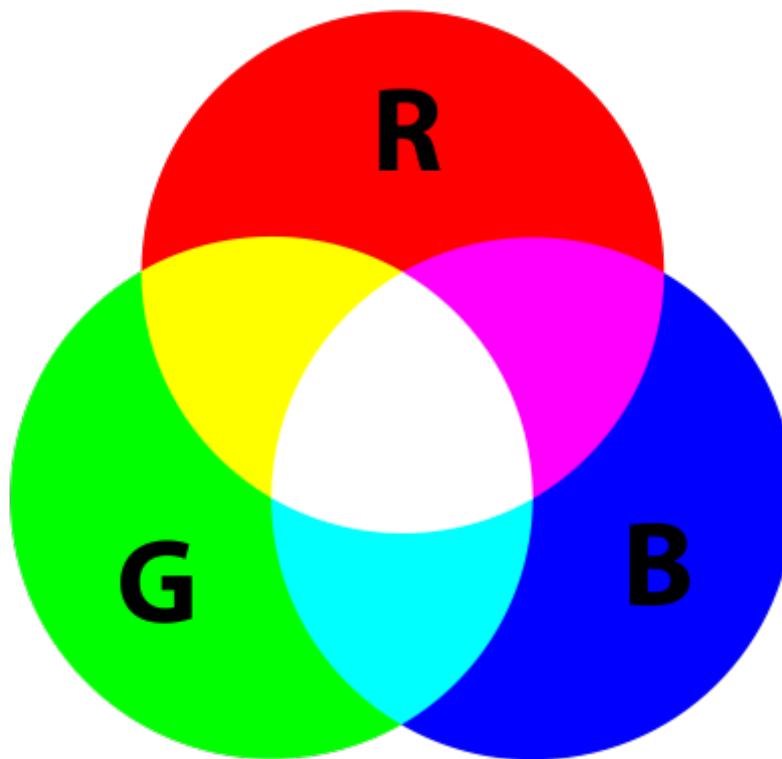


Figure 3: Additieve kleuren

Om wit te krijgen, heb je alledrie de kleuren nodig.

## Vragen

- 1. Welke drie kleuren lampjes heeft een pixel?
- 2. Met welke lampjes samen maak je geel?
- 3. Met welke lampjes samen maak je cyaan/lichtblauw?
- 4. Met welke lampjes samen maak je magenta/paars?

- 5. Met welke lampjes samen maak je wit?
- 6. Met welke lampjes samen maak je zwart?
- 7. Met welke lampjes samen maak je grijs?
- 8. Met welke lampjes samen maak je oranje?

## Antwoorden

- 1. Rood, groen en blauw
- 2. Rood en groen
- 3. Groen en blauw
- 4. Rood en blauw
- 5. Rood en groen en blauw
- 6. Met geen lampjes: als alle lampjes uit zijn, is het zwart
- 7. Met rood en groen en blauw, maar dan moeten ze niet op hun hardst branden
- 8. Met rood op z'n hardst en groen op halve kracht

## background

In Processing is er een functie om de achtergrond een kleur te geven. Deze functie heet **background**. **background** is Engels voor 'achtergrond'. **background** is een functie die drie getallen nodig heeft. Deze drie getallen bepalen hoe hard de rode, groene en blauwe lampjes gaan schijnen. Deze drie getallen noemen we de RGB waarde. Met het getal nul zeg je dat een lampje uit staat. Met het getal 255 zeg je dat een lampje aan staat. Met getallen tussen nul en 255 kun je het lampje ertussenin laten branden.

Met deze Processing code krijg je een rode achtergrond:

```
void setup()
{
  size(100,100);
}

void draw()
{
  background(255,0,0);
}
```



## Opdracht

- 1. Kopieer deze code in Processing en start de code
- 2. Wat is een RGB waarde?
- 3. Wat is de RGB waarde van groen? Maak in Processing een groene achtergrond
- 4. Wat is de RGB waarde van blauw? Maak in Processing een blauwe achtergrond
- 5. Wat is de RGB waarde van geel? Maak in Processing een gele achtergrond
- 6. Wat is de RGB waarde van cyaan/lichtblauw? Maak in Processing een cyane/lichtblauwe achtergrond
- 7. Wat is de RGB waarde van magenta/paars? Maak in Processing een magenta/paarse achtergrond
- 8. Wat is de RGB waarde van wit? Maak in Processing een witte achtergrond
- 9. Wat is de RGB waarde van zwart? Maak in Processing een zwart achtergrond
- 10. Wat is de RGB waarde van grijs? Maak in Processing een grijze achtergrond
- 11. Wat is de RGB waarde van donkerrood? Maak in Processing een donkerrode achtergrond
- 12. Wat is de RGB waarde van oranje? Maak in Processing een oranje achtergrond

## Oplossingen

- 1. OK
- 2. De Rood-Groen-Blauw waarde. Dit zijn drie getallen van nul tot en met 255 die bepalen hoe hard de drie kleurenlampjes branden
- 3. `background(0,255,0)`
- 4. `background(0,0,255)`
- 5. `background(255,255,0)`
- 6. `background(0,255,255)`
- 7. `background(255,255,0)`
- 8. `background(255,255,255)`

- 9. `background(0,0,0)`
- 10. `background(128,128,128)`, maar andere getallen tussen 0 en 255 zijn ook goed. Als ze maar alledrie gelijk zijn
- 11. `background(128,0,0)`, maar het eerste getal mag ook een ander getal tussen de 0 en 255 zijn
- 12. `background(255,128,0)`, maar het tweede getal mag ook een ander getal in de buurt van 128 zijn

## Bal naar rechts

In deze les gaan we een bal naar rechts laten bewegen.

Het ziet er zo uit:

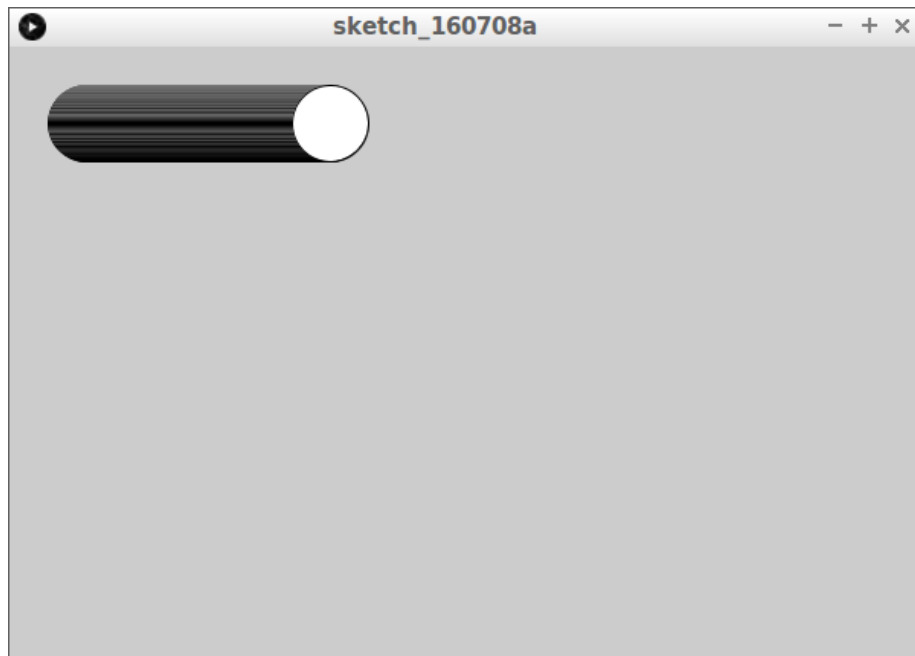


Figure 1: Bal naar rechts

We leren in deze les wat een variabele is. Je kunt (bijna) niet programmeren zonder variabelen.

### Wat weten we al?

Als je de vorige lessen hebt gedaan, weet je wat deze code doet:

```
void setup()
{
  size(600, 400);
}

void draw()
{
  ellipse(50,50,50,50);
}
```

## Vragen

- Wat doet dit programma?
- Waar wordt deze cirkel getekend? Komt deze cirkel tegen de rand aan?

We gaan de code aanpassen:

```
float x = 50;

void setup()
{
  size(600, 400);
}

void draw()
{
  ellipse(x,50,50,50);
}
```

## Vragen

- Wat doet dit programma?
- Wat zijn de verschillen?

De eerste nieuwe regel is:

```
float x = 50;
```

In mensentaal is dit: ‘Lieve computer, onthoud het getal x. x heeft een beginwaarde van vijftig.’

Een variabele is iets dat onthouden moet worden. Een kassa onthoudt bijvoorbeeld de hoeveelheid geld die alle boodschappen bij elkaar zijn. Variabelen die jij weet, zijn: je naam, je leeftijd, je geboortedatum, je adres, je telefoonnummer, je emailadres, en nog veel meer. Als iemand je je leeftijd vraagt, dan weet je welk getal je moet zeggen.

Het woord **x** is de naam van een variable. In dit geval van hoe ver de cirkel naar rechts staat. Het woord **float** betekent dat ‘x’ een getal is. Het symbool = betekent ‘wordt vanaf nu’. Het getal 50 is de beginwaarde.

De tweede veranderde regel is:

```
ellipse(x,50,50,50);
```

In mensentaal is dit: ‘Lieve computer, teken een ovaal die:

- x naar rechts is. De computer weet nog wel wat x is: vijftig!
- 50 omlaag is
- 50 pixels breed is

- 50 pixels hoog is

### Vragen

- Wat als je `float x = 50;` weghaalt?
- Wat als je `float x = 50;` verandert naar `float rechts = 50;`?
- Wat als je `float x = 50;` verandert naar `float x = 100;`?
- Wat als je `ellipse(x,50,50,50);` weghaalt?
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(rechts,50,50,50);`?
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,50,50);`?
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,50);`?
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,x);`?
- Wat als je `x` vervangt door `rechts`?
- Wat als je `x` vervangt door `dinosaurus`?

Nu gaan we de cirkel laten bewegen:

```
float x = 50;
```

```
void setup()
{
  size(600, 400);
}
```

```
void draw()
{
  ellipse(x,50,50,50);
  x = x + 1;
}
```

### Vragen

- Wat doet dit programma?
- Wat zijn de verschillen?

De nieuwe regel is:

```
x = x + 1;
```

In mensentaal is dit: ‘Lieve computer, x is vanaf nu x plus een’. Of: ‘Maak x een hoger’.

### Vragen

- Als x vijftig is, wat is x dan na `x = x + 1;`?
- Als x 51 is, wat is x dan na `x = x + 1;`?

- Als x 52 is, wat is x dan na `x = x + 1;`?
- Als x 53 is, wat is x dan na `x = x + 1;`?
- Als x 54 is, wat is x dan na `x = x + 1;`?

Nu kunnen we snappen wat het programma doet. Hier staat het programma weer:

```
float x = 50;

void setup()
{
  size(600, 400);
}

void draw()
{
  ellipse(x,50,50,50);
  x = x + 1;
}
```

De eerste keer dat de computer **draw** gaat doen, dan vult deze voor **x** een 50 in. Daarna wordt x een hoger. Dan is **draw** klaar.

De tweede keer dat de computer **draw** gaat doen, dan vult deze voor **x** een 51 in. Daarna wordt x een hoger. Dan is **draw** klaar.

De derde keer dat de computer **draw** gaat doen, dan vult deze voor **x** een 52 in. Daarna wordt x een hoger. Dan is **draw** klaar.

## Vragen

- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,50,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,50,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,50,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,50,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,50,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,x,x);`?
- Wat als je `x = x + 1;` vervangt door `x = x + 2;`?
- Wat als je `x = x + 1;` vervangt door `x = x + 10;`?
- Wat als je `x = x + 1;` vervangt door `x = x + 0;`?

- Wat als je  $x = x + 1$ ; vervangt door  $x = x - 1$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x - 0$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x * 2$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x * 1$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x * 0$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x / 2$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x / -2$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x / 1$ ;
- Wat als je  $x = x + 1$ ; vervangt door  $x = x / 0$ ;

## Verder

Je zou nu kunnen doen:

- Bal die eeuwig naar rechts gaat

## Bal die eeuwig naar rechts gaat

In deze les gaan we een bal eeuwig naar rechts laten gaan.

Het ziet er zo uit:



Figure 1: Bal die eeuwig naar rechts gaat 1



Figure 2: Bal die eeuwig naar rechts gaat 2

We leren in deze les wat `if`-statement is. Je kunt (bijna) niet programmeren zonder `if`-statements.

### Wat weten we al?

Als je de vorige lessen hebt gedaan, weet je wat deze code doet:

```
float x = 50;

void setup()
{
  size(600, 400);
}

void draw()
{
  ellipse(x,50,50,50);
  x = x + 1;
}
```



## Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal
- Blijft de ovaal zichtbaar op het scherm?
- Kopieer de code en bekijk het programma. Klopt wat je dacht?

## Een if-statement

We willen kunnen zeggen: ‘Lieve computer, *als* de bal te ver naar rechts is, dan teleporteer je de bal naar rechts’. **if** is Engels voor ‘als’.

Zo zou dit kunnen:

```
if (x > 200)
{
    x = 100;
}
```

Dit betekent:

- **if**: begin van een if statement. Een if-statement heeft dan twee gedeeltes:
- **()**: tussen de ronde haken staat een test; iets wat waar of niet waar is
- **{}**: tussen de accolades staat wat de computer moet doen als de test waar is
- **x > 200**: dit staat tussen de ronde haken. Dit is de test ‘x is groter dan 200’. Het > tekenje betekent ‘groter dan’
- **x = 100**: dit staat tussen de accolades. Als ‘x is groter dan 200’ waar is, dan krijgt x de waarde 100

Preciezer zeg je: ‘Lieve computer, *als* x meer is dan 200, zet x dat op 100’. **if** is Engels voor ‘als’.

## Vragen

- Kopieer het **if**-statement tussen de accolades van de **draw** functie
- Wat doet het programma?

Als het kopiëren niet is gelukt, gebruik dan deze code:

```
float x = 50;

void setup()
{
    size(600, 400);
}
```

```

void draw()
{
  ellipse(x,100,100,100);
  x = x + 1;
  if (x > 200)
  {
    x = 100;
  }
}

```

- Kun je ervoor zorgen dat de ovaal helemaal naar de linkerkant van het scherm springt?
- Kun je ervoor zorgen dat de ovaal helemaal naar rechts beweegt, voordat deze naar de linkerkant van het scherm springt?

## Antwoord

Dit is een eeuwig naar rechts gaande bal:

```
float x = -50;
```

```

void setup()
{
  size(600, 100);
}

```

```

void draw()
{
  ellipse(x,50,100,100);
  x = x + 1;
  if (x > 650)
  {
    x = -50;
  }
}

```

Het lijkt al een beetje op Lonelier Pong. Dit is geen toeval :-)

## Verder

Je zou nu kunnen doen:

- Bal die horizontaal stuitert

## Bal die horizontaal stuitert

In deze les gaan we een bal horizontaal laten stuiten.

Het ziet er zo uit:

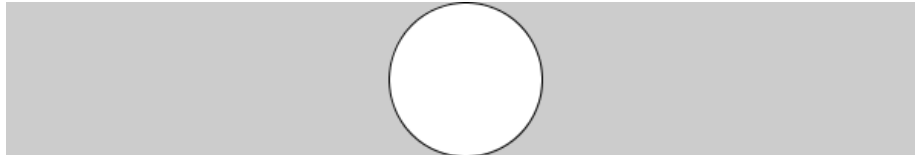


Figure 1: Bal die horizontaal stuitert (zie 'dojo/Images/BalDieHorizontaalStuitertGif')

We gaan in deze les twee variabelen en twee `if`-statements gebruiken.

### Wat weten we al?

Dit is een eeuwig naar rechts gaande bal:

```
float x = 300;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x, 50, 100, 100);
  x = x + 1;
  if (x > 650)
  {
    x = -50;
  }
}
```

### Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal?
- Blijft de ovaal zichtbaar op het scherm?
- Kopieer de code en bekijk het programma. Klopt wat je dacht?
- Verander het programma: laat de bal nu eeuwig naar links gaan

## Twee variabelen

Nu onthoudt de computer een variabele: de x-coördinaat van de ovaal. Om een bal te laten stuiteren, moet er ook een richting onthouden worden: de bal gaat immers of naar links of naar rechts.

In deze code wordt de richting van de bal **dx** genoemd. Dit is een afkorting van ‘delta x’ en dat is weer wiskundetaal voor ‘de verandering van x’.

## Vragen

```
float x = 300;
float dx = 2;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    x = -50;
  }
}
```

- In welke richting beweegt de bal?
- Hoeveel pixels beweegt de bal per keer?
- Zet de waarde van **dx** op 1. Wat zie je?
- Zet de waarde van **dx** op 0. Wat zie je?
- Zet de waarde van **dx** op -1. Wat zie je?
- Bij sommige waarden van **dx** gaat de bal links het beeld uit. Maak een tweede if-statement, die ervoor zorgt dat de bal eeuwig links kan gaan. Test dit bij een **dx** van 2, 0 en -2.
- Wat moet er met de **dx** gebeuren om de bal te laten stuiteren? Probeer dit!

## Stuiteren

Als een bal met een snelheid van drie pixels naar rechts gaat en stuitert, dan gaat deze vanaf dan drie pixels naar links. Andersom is dat ook zo.

Hier is een manier om de bal te laten stuiteren:

```
float x = 300;
float dx = 2;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    dx = -2;
  }
  if (x < 50)
  {
    dx = 2;
  }
}
```

## Vragen

- Kopieer en run de code. Stuitert de bal?
- Verander de snelheid van de bal naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal naar drie pixels per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal terug naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen? Laat nu de bal in het begin naar links gaan. Op hoeveel plekken moet je de code aanpassen?

## De richting omklappen

Er is een slimmere manier om `dx` te veranderen. We hebben gezien dat als `dx` gelijk was aan 2, deze -2 wordt bij bij een stuiters. We hebben gezien dat als `dx` gelijk was aan -2, deze 2 wordt bij bij een stuiters. Er komt een minnetje voor, of er komt een minnetje bij. Dit is gemakkelijk op dezelfde manier te doen:

```
dx = -dx;
```

Hiermee zeg je ‘Lieve computer, de nieuwe waarde van `dx` is min de oude waarde’. Als de oude waarde van `dx` 2 is, dan wordt deze nu -2. Als de oude waarde van `dx` -2 is, dan wordt deze nu --2 (jawel, min min twee) en dat mag je schrijven als 2 (omdat min keer min is plus).

## Opdracht

- Gebruik nu de slimme manier om een bal te laten stuiteren.

Het lijkt al een beetje op Lonelier Pong. Dit is geen toeval :-)

## Zwaartekracht

In deze les gaan we zwaartekracht programmeren.

Het ziet er zo uit:

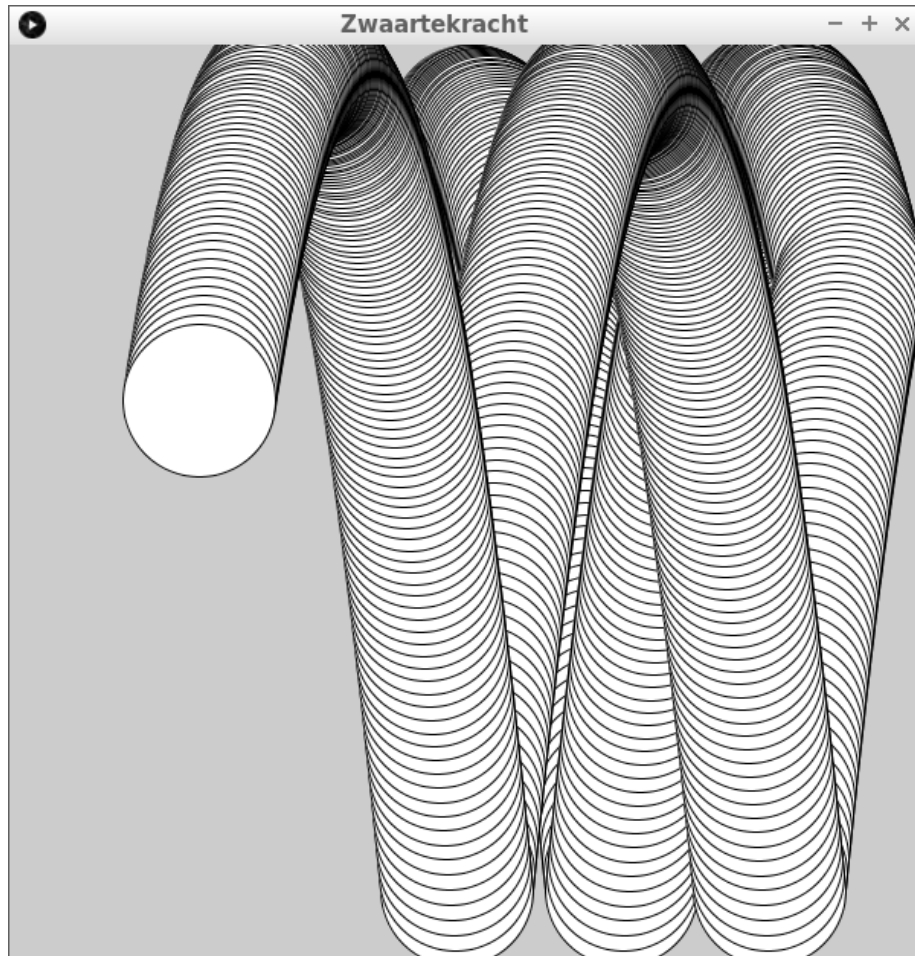


Figure 1: Zwaartekracht

We gaan in deze les twee variabelen en twee `if`-statements gebruiken.

### Wat weten we al?

Dit is een eeuwig horizontaal stuitende bal:

```
float x = 300;
```

```

float dx = 1; //Snelheid in de x richting

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 550 || x < 50)
  {
    dx = -dx;
  }
}

```

## Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal?
- Blijft de ovaal zichtbaar op het scherm?

## Opdrachten

- Laat de bal nu *ook* verticaal bewegen. Gebruik de variabele namen y en dy. Zorg dat de bal ook verticaal stuitert

## Zwaartekracht

De zwaartekracht trekt aan voorwerpen. Iets dat omlaag valt, gaat hierdoor steeds sneller vallen. Iets dat omhoog gaat, gaat eerst steeds langzamer omhoog, en gaat daarna ook vallen. In de natuukunde gebruiken ze **g** (van ‘gravity’, dit is Engels voor zwaartekracht) voor de zwaartekracht.

## Opdacht

In onze code hebben we nu een dy. Dit is de snelheid in de y richting. Elke beurt moet dy nu **g** groter worden. Een mooie waarde voor **g** is 0.1.

Voeg toe dat de cirkel op een natuurlijke manier omlaag valt.



## Twee variabelen

Nu onthoudt de computer een variabele: de x-coördinaat van de ovaal. Om een bal te laten stuiteren, moet er ook een richting onthouden worden: de bal gaat immers of naar links of naar rechts.

In deze code wordt de richting van de bal **dx** genoemd. Dit is een afkorting van ‘delta x’ en dat is weer wiskundetaal voor ‘de verandering van x’.

## Vragen

```
float x = 300;
float dx = 2;

void setup()
{
    size(600, 100);
}

void draw()
{
    ellipse(x,50,100,100);
    x = x + dx;
    if (x > 650)
    {
        x = -50;
    }
}
```

- In welke richting beweegt de bal?
- Hoeveel pixels beweegt de bal per keer?
- Zet de waarde van **dx** op 1. Wat zie je?
- Zet de waarde van **dx** op 0. Wat zie je?
- Zet de waarde van **dx** op -1. Wat zie je?
- Bij sommige waarden van **dx** gaat de bal links het beeld uit. Maak een tweede if-statement, die ervoor zorgt dat de bal eeuwig links kan gaan. Test dit bij een **dx** van 2, 0 en -2.
- Wat moet er met de **dx** gebeuren om de bal te laten stuiteren? Probeer dit!

## Stuiteren

Als een bal met een snelheid van drie pixels naar rechts gaat en stuitert, dan gaat deze vanaf dan drie pixels naar links. Andersom is dat ook zo.

Hier is een manier om de bal te laten stuiten:

```
float x = 300;
float dx = 2;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    dx = -2;
  }
  if (x < 50)
  {
    dx = 2;
  }
}
```

## Vragen

- Kopieer en run de code. Stuitert de bal?
- Verander de snelheid van de bal naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal naar drie pixels per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal terug naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen? Laat nu de bal in het begin naar links gaan. Op hoeveel plekken moet je de code aanpassen?

## De richting omklappen

Er is een slimmere manier om `dx` te veranderen. We hebben gezien dat als `dx` gelijk was aan 2, deze -2 wordt bij bij een stuit. We hebben gezien dat als `dx` gelijk was aan -2, deze 2 wordt bij bij een stuit. Er komt een minnetje voor, of er komt een minnetje bij. Dit is gemakkelijk op dezelfde manier te doen:

```
dx = -dx;
```

Hiermee zeg je ‘Lieve computer, de nieuwe waarde van `dx` is min de oude waarde’. Als de oude waarde van `dx` 2 is, dan wordt deze nu -2. Als de oude waarde van `dx` -2 is, dan wordt deze nu --2 (jawel, min min twee) en dat mag je schrijven als 2 (omdat min keer min is plus).

## Opdracht

- Gebruik nu de slimme manier om een bal te laten stuiteren.

Het lijkt al een beetje op Lonelier Pong. Dit is geen toeval :-)

## text

Tekst wordt veel gebruikt, ook in games, voor bijvoorbeeld een score.

Hier zie je 'Zork, the underground empire', een van de beroemdste tekstavonturen ooit:

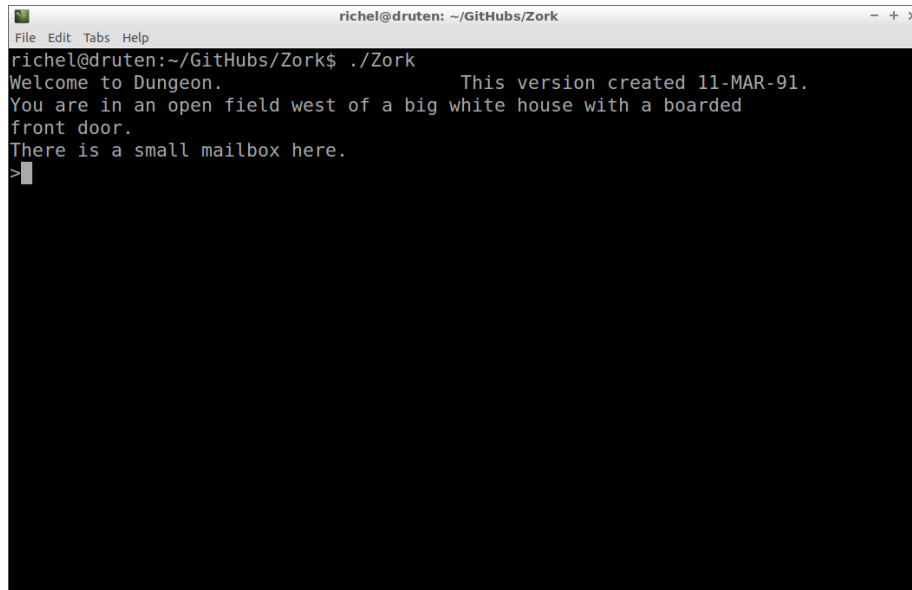
A screenshot of a terminal window titled 'richel@druten: ~/GitHubs/Zork'. The window contains the following text: 'richel@druten:~/GitHubs/Zork\$ ./Zork', 'Welcome to Dungeon.', 'This version created 11-MAR-91.', 'You are in an open field west of a big white house with a boarded front door.', 'There is a small mailbox here.', and a prompt '>' with a cursor. The terminal has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'.

Figure 1: Zork

In deze les gaan we leren

- hoe je tekst op het scherm zet
- hoe je berekeningen op het scherm zet
- hoe je tekst vergroot
- hoe je tekst een kleur geeft

Zo gaat het eruit zien:

Kun je nog geen puntjes tekenen? Ga dan naar de les waarin je puntjes leert tekenen

Kun je nog geen vlakken inkleuren? Ga dan naar de les 'fill'

## Tekst

Hier zie je de tekst 'Hallo' laat zetten op coördinaat (10,20):

```
text("Hallo", 10, 20);
```

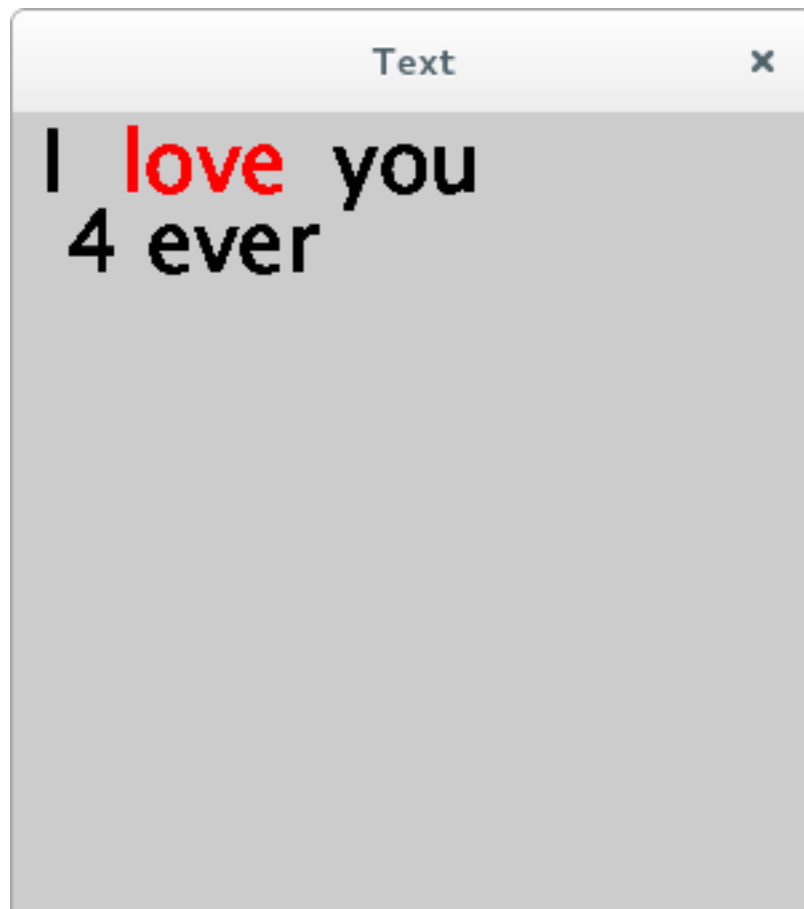


Figure 2: Text

Let op dat de tekst tussen dubbele apostroffen (") moet.

`text` kan ook rekenen!

Hier plus en min:

```
text(128 + 64, 10, 20);  
text(128 - 64, 10, 20);
```

Hier een keersom:

```
text(16 * 16, 10, 20);
```

Hier een deelsom:

```
text(256 / 16, 10, 20);
```

Tekstgrootte kun je aanpassen met

```
textSize(32);
```

Tekstkleur kun je aanpassen met `fill`:

```
fill(255, 0, 0);
```

## Opdracht

Zet de tekst `I love you 4 ever` op het scherm, waarbij:

- alle woorden zwart zijn, behalve `love`, die rood is
- de `4` is de uitkomst van een berekening, bijvoorbeeld `2 + 2` (maar hoe moeilijker de berekening, hoe stoerder)

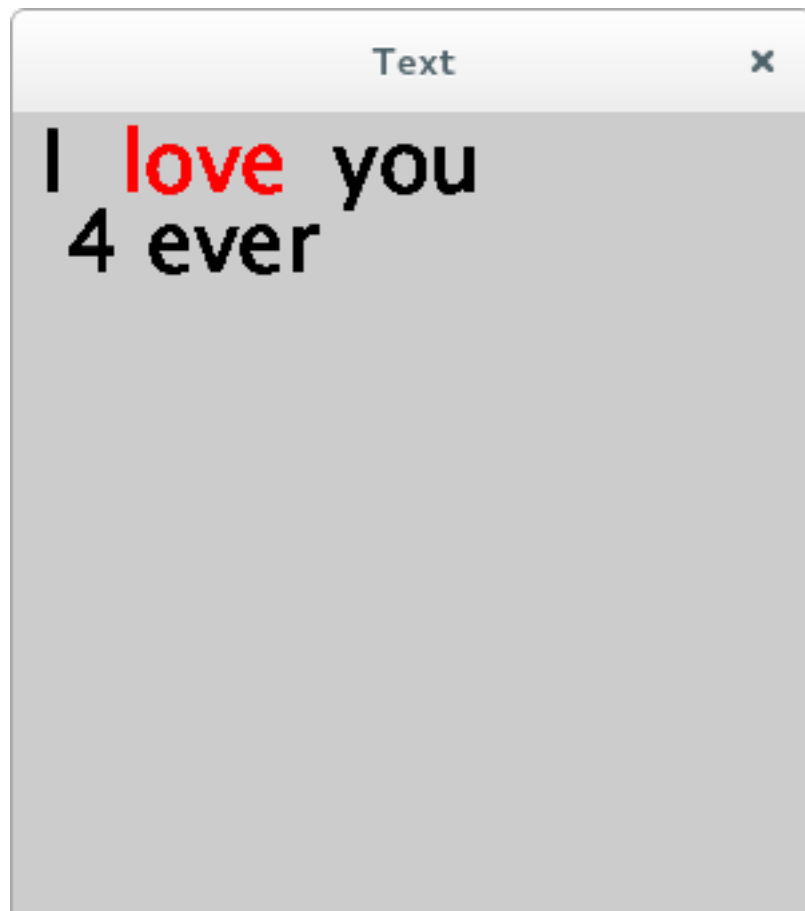


Figure 3: Text