

Figure 1: Boek 1

#	Omschrijving
1	Een mooi programma
2	Bal naar rechts
3	width en height
4	point en random

# Contents

Voorwoord	1
Een Mooi Programma	2
Bal naar rechts	6
width en height	25
point en random	33

## Voorwoord

Dit is een boek over Processing, geschreven voor jonge tieners. Processing is een programmeertaal. Dit boek leert je die programmeertaal.

### Over dit boek

Dit boek heeft een CC-BY-NC-SA licentie.



Figure 1: De licentie van dit boek

(C) Richèl Bilderbeek en alle docenten en alle leerlingen, 2016

Met dit boekje mag je alles doen wat je wilt, als je maar verwijst naar de oorspronkelijke versie op deze website: [https://github.com/richelbilderbeek/processing\\_voor\\_jonge\\_tieners](https://github.com/richelbilderbeek/processing_voor_jonge_tieners). Dit boekje zal altijd gratis, vrij en open blijven.

Het is nog een beetje een slordig boek. Er zitten tiepvauten in en de opmaak is **niet altijd even mooi**. Omdat dit boek op een website staat, kan iedereen die dit boek te slordig vindt minder slordig maken.

# 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:

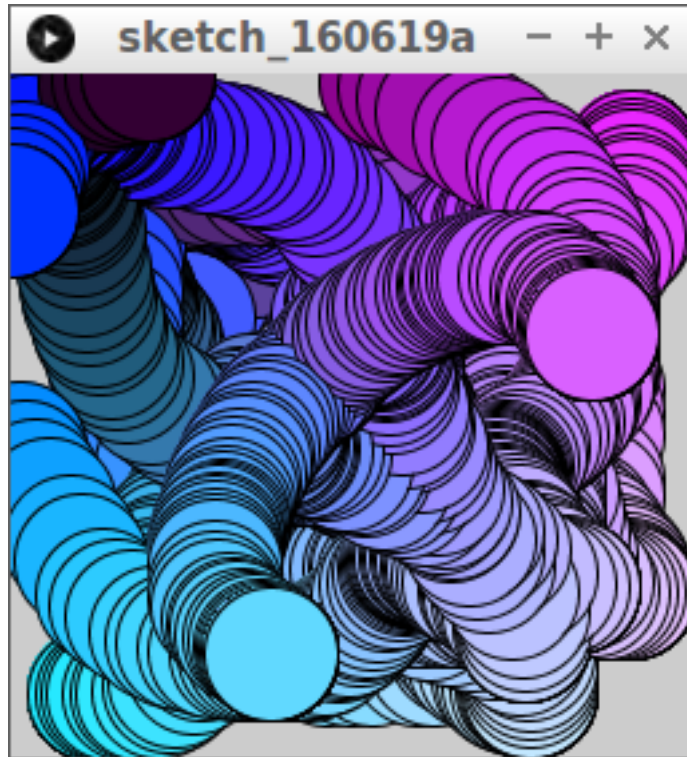


Figure 2: EenMooiProgramma

## Een mooi programma: intro

Processing begint met een leeg programma zonder code:

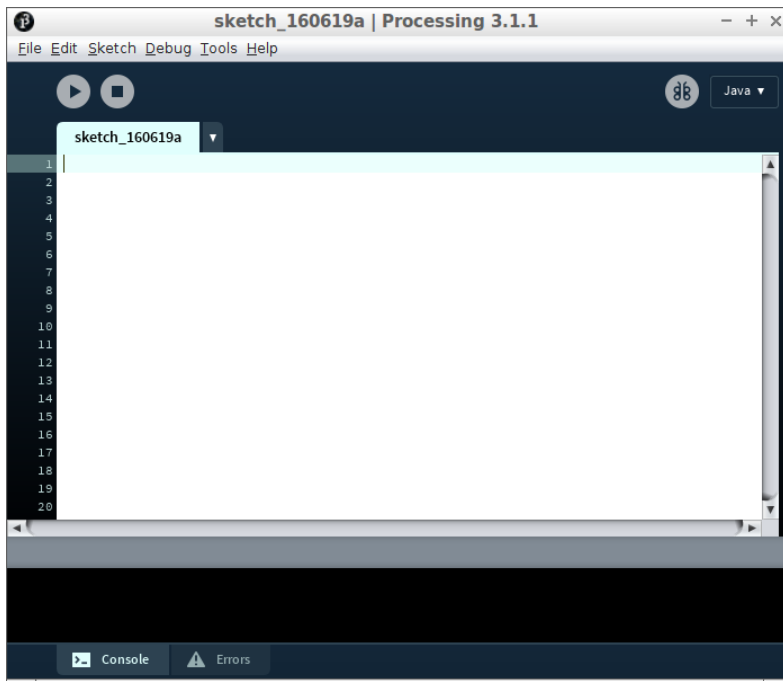


Figure 3: Processing zonder code

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.

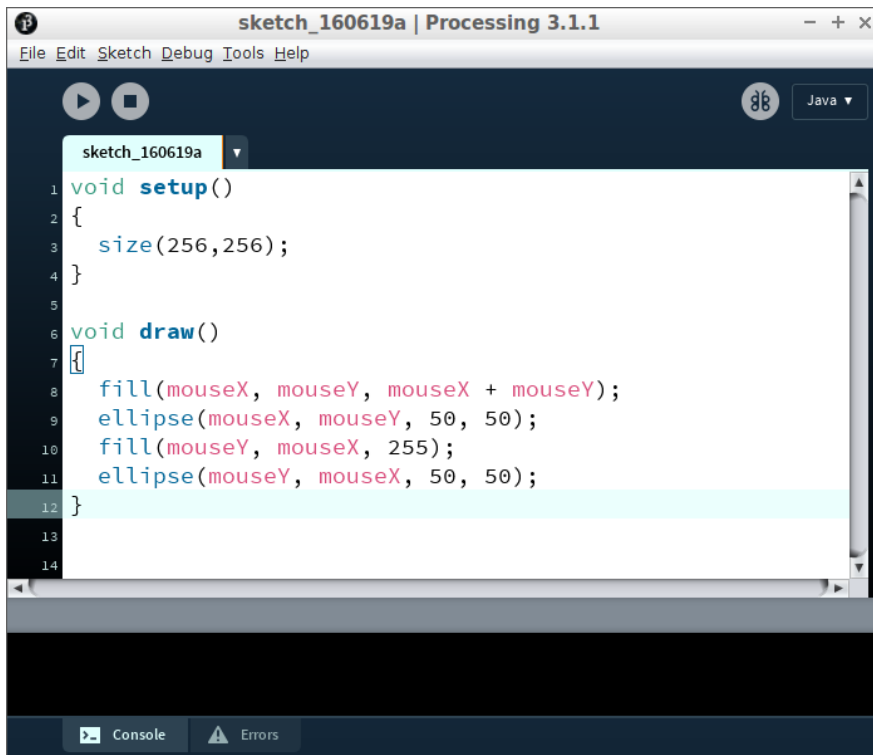


Figure 4: Processing met code

## Een mooi programma: eindopdracht

- Start Processing
- Run deze code, door op de 'Run' knop te klikken

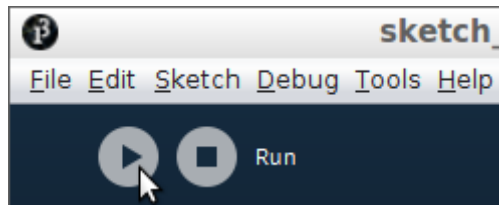


Figure 5: De Run knop



Gelukt? Laat dit zien aan een volwassene voor een sticker!

---

---

## Links

- Een Mooi Programma: YouTube, mp4

## Bal naar rechts

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

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



Figure 6: Marble Madness

## Bal naar rechts: intro

Type de volgende code over:

```
float x = 60;

void setup()
{
  size(250, 200);
}

void draw()
{
  ellipse(x, 50, 40, 30);
  x = x + 1;
}
```

Druk dan op 'Run' (zie figuur Druk op 'Run').

Als er rode letters komen, heb je een typefout gemaakt. Kijk goed en verbeter de typefouten.

Als alles goed gaat, zie je een bal die naar rechts beweegt (zie figuur Bal naar rechts: intro).

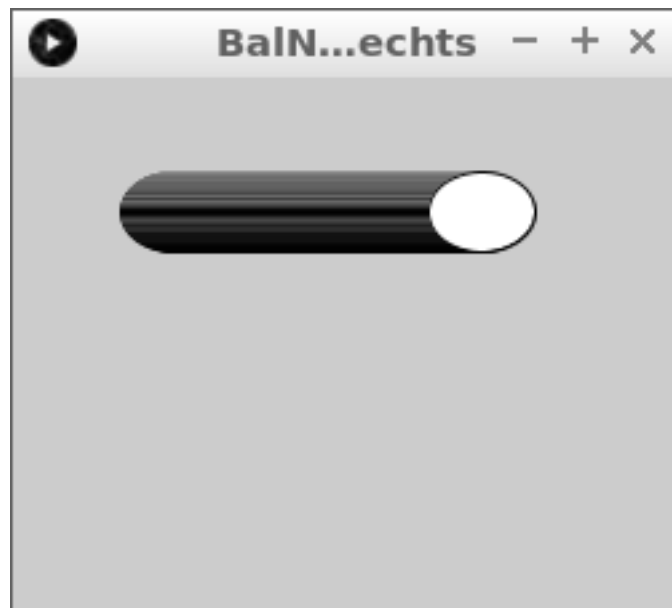


Figure 7: Bal naar rechts: intro



## Bal naar rechts: opdracht 1

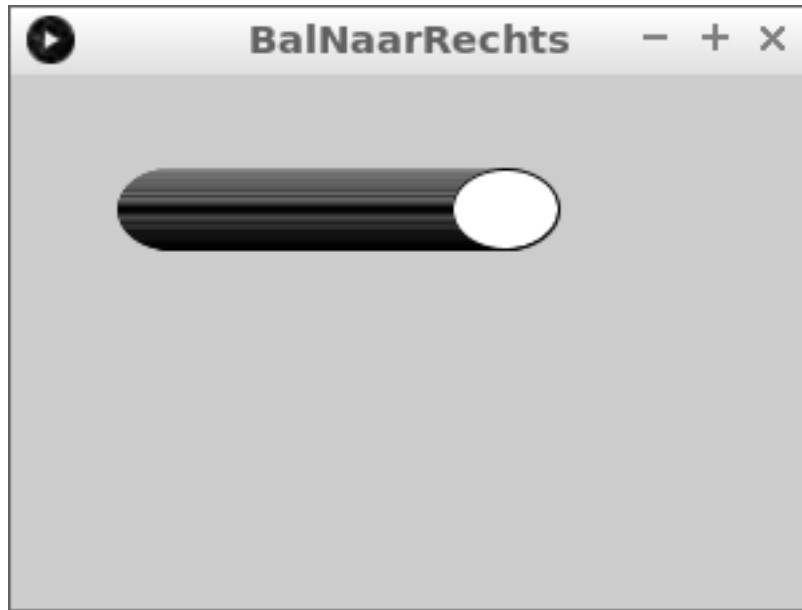


Figure 8: Bal naar rechts: opdracht 1

Het scherm is nu 250 pixels breed. Maak deze nu 300 pixels breed.

Verander de code en druk op 'Run'.

## Bal naar rechts: oplossing 1

Er zit een 250 in de code. Deze naar 300 veranderen is genoeg:

```
float x = 60;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, 50, 40, 30);
  x = x + 1;
}
```



---

`size(300, 200);`

‘Lieve computer, maak een venster van 300 pixels breed en 200 pixels hoog.’

---

## Bal naar rechts: opdracht 2

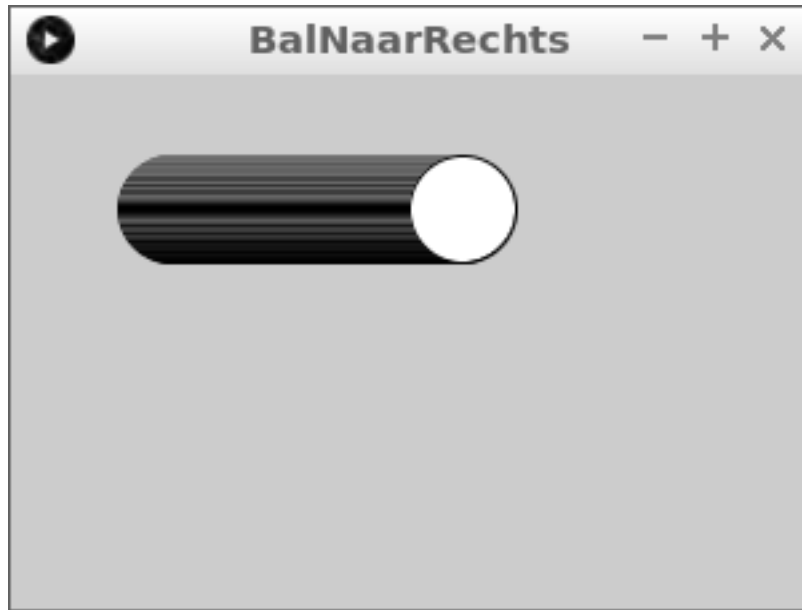


Figure 9: Bal naar rechts: opdracht 2

De bal is nu ei-vorming: hij is nu 40 pixels breed en 30 pixels hoog. Maak de bal nu rond: 40 pixels breed en 40 pixels hoog.

## Bal naar rechts: oplossing 2

`ellipse(x, 50, 40, 30);` tekent de bal. De 40, 30 zorgt ervoor dat de bal eiv-ormig is. Door dit 40, 40 te maken, wordt de bal rond.

```
float x = 60;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 1;
}
```



---

`ellipse(x,50,40,30);`

‘Lieve computer, teken een ovaal x pixels naar rechts, 50 pixels omlaag, die 40 pixels breed en 30 pixels hoog is.’

---

## Bal naar rechts: opdracht 3

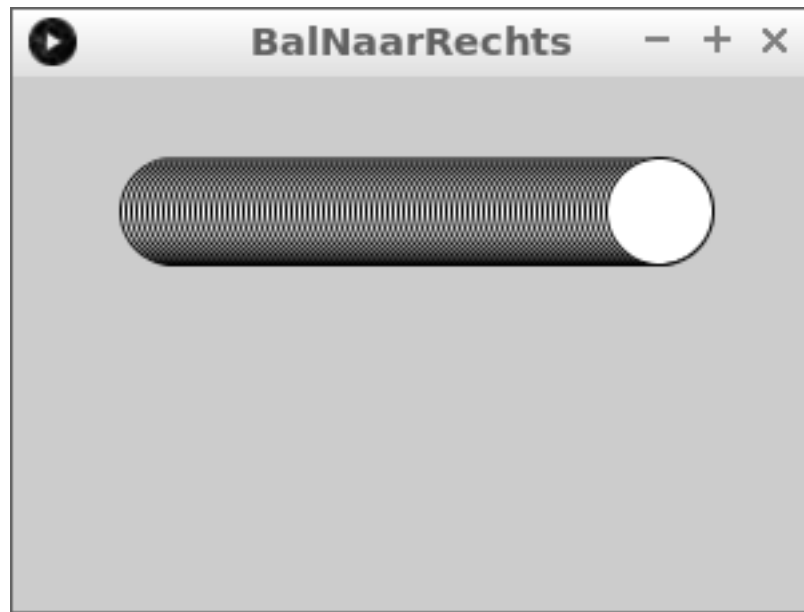


Figure 10: Bal naar rechts: opdracht 3

De bal gaat nu met een snelheid van 1 pixel per keer naar rechts. Laat de bal twee keer zo snel naar rechts gaan

## Bal naar rechts: oplossing 3

$x = x + 1$ ; beweegt de bal. Verander dit naar  $x = x + 2$ ;. De code wordt dan:

```
float x = 60;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 2;
}
```



---

$x = x + 1$ ;	'Lieve computer, maak x een hoger.'
$x += 1$ ;	'Lieve computer, maak x een hoger.'
$x++$ ;	'Lieve computer, maak x een hoger.'
$++x$ ;	'Lieve computer, maak x een hoger.'

---

## Bal naar rechts: opdracht 4

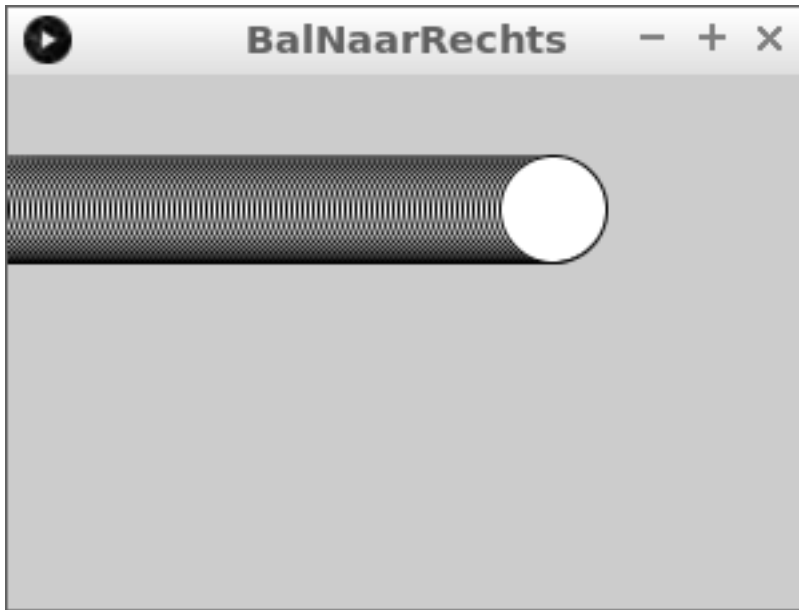


Figure 11: Bal naar rechts: opdracht 4

In het begin zit het midden van de bal 60 pixels naar rechts. Kun je de cirkel ook 0 pixels naar rechts laten beginnen?

## Bal naar rechts: oplossing 4

`float x = 60;` bepaalt dit. Verander dit naar `float x = 0;`. De code wordt dan:

```
float x = 0;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 2;
}
```



---

```
void setup() { } 'Lieve computer, doe wat tussen accolades staat een keer.'
```

---



## Bal naar rechts: opdracht 5

Haha, deze les heet ‘Bal naar rechts’, toch gaan we ook een bal naar links laten bewegen!

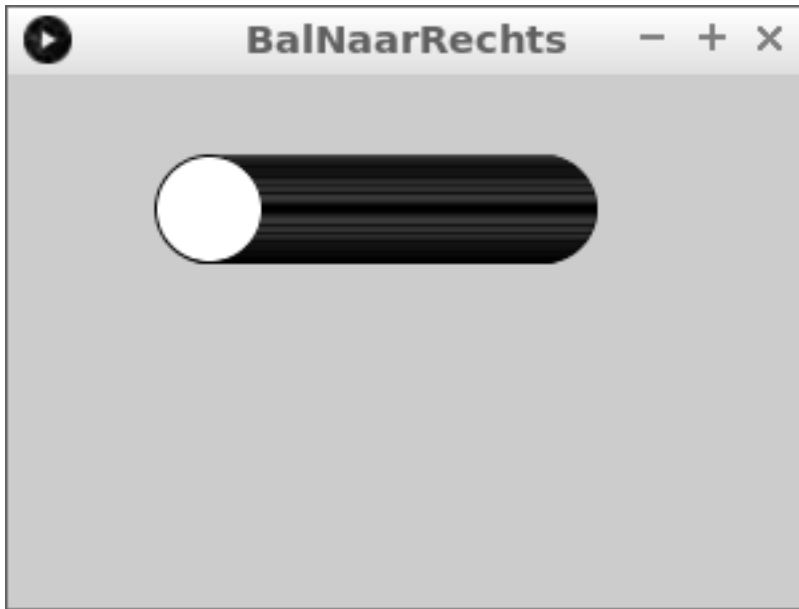


Figure 12: Bal naar rechts: opdracht 5

Laat de bal nu aan de rechterkant van het scherm beginnen en naar links gaan.

## Bal naar rechts: oplossing 5

Om de bal aan de rechtekant te krijgen moet je `float x = 500;` gebruiken (of een ander hoog getal). Om de bal naar links te laten bewegen, moet je `x = x - 1;` gebruiken. De code wordt dan:

```
float x = 200;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x - 1;
}
```



---

```
void draw() { }
```

‘Lieve computer, doe de hele tijd wat tussen accolades staat.’

---



## Bal naar rechts: wat is een variabele?

In de eerste regel gebruiken we een variabele:

```
float x = 50;
```

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

---

	
<code>float x = 50;</code>	‘Lieve computer, onthoud het getal $x$ met een beginwaarde van vijftig.’

---

Een variabele is een stukje computergeheugen met een naam. De computer kan aan die naam bepalen waar in het geheugen hij moet kijken.

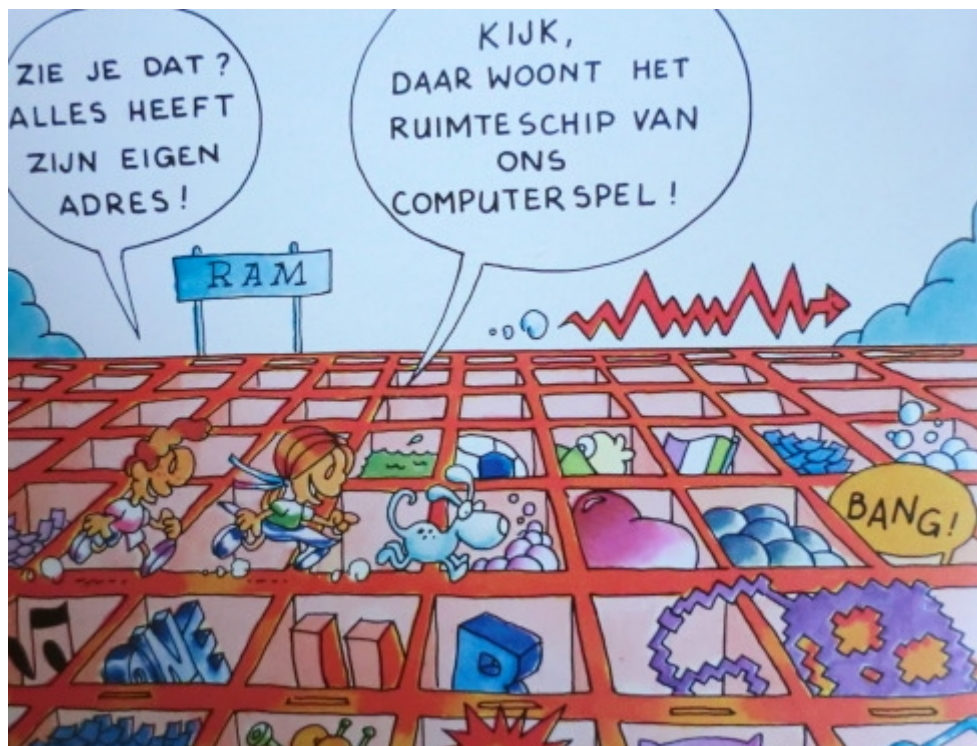




Figure 13: Het geheugen van een computer

Variabelen die bij jou (en bijna elk mens) horen, zijn: naam, leeftijd, geboortedatum, adres, telefoonnummer, emailadres, en nog veel meer. Als iemand je je leeftijd vraagt, dan weet je welk getal je moet zeggen.

---

	
<code>geld</code> <code>1000000</code>	‘Lieve computer, zeg hoeveel geld ik op de bank heb.’

---

Terug naar de eerste regel van onze code:

```
float x = 50;
```

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 (komma)getal is. Het symbool `=` betekent 'wordt vanaf nu'. Het getal `50` is de beginwaarde. De puntkomma (`;`) geeft het einde van een zin aan (zoals de punt in een Nederlandse tekst).

1

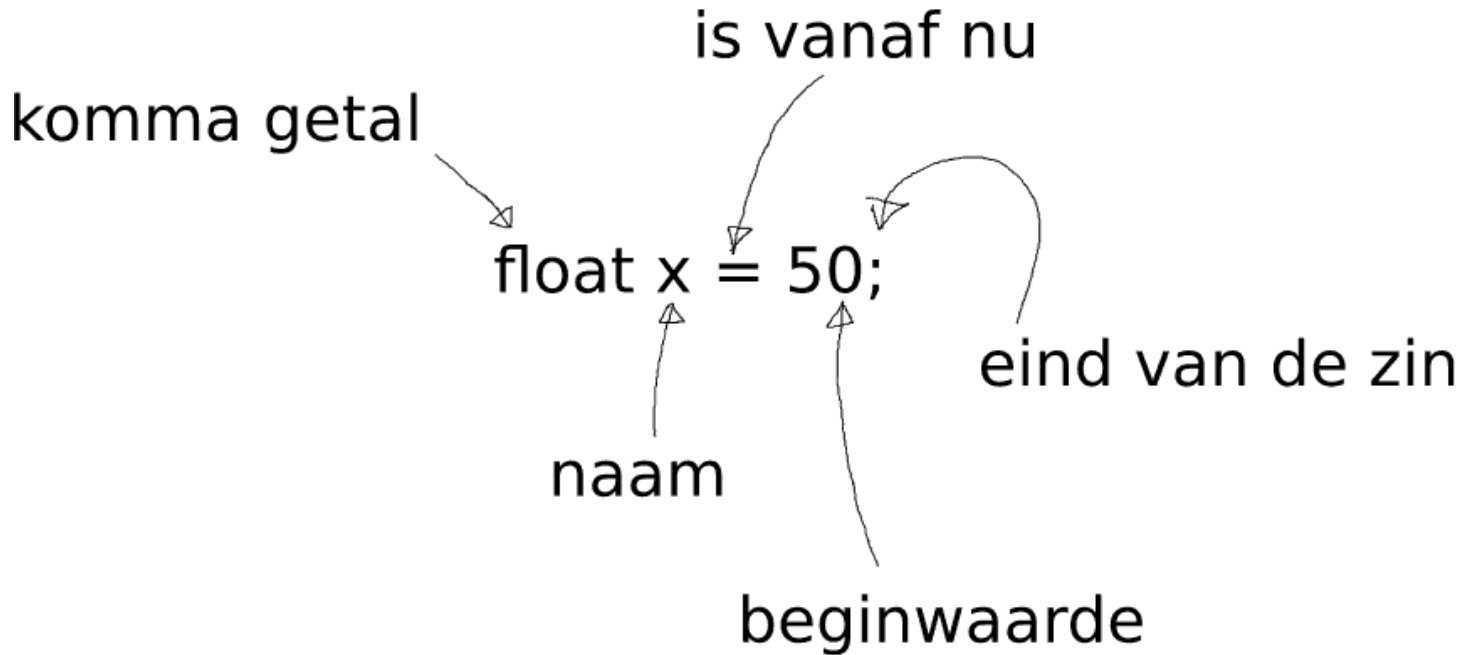




Figure 14: Uitleg van `float x = 50;`

	
<code>float</code>	'Een komma getal'
<code>=</code>	'is vanaf nu'
<code>;</code>	'.'

## Bal naar rechts: opdracht 6

Haha, deze les heet ‘Bal naar rechts’, toch gaan we ook een bal naar onder laten bewegen!

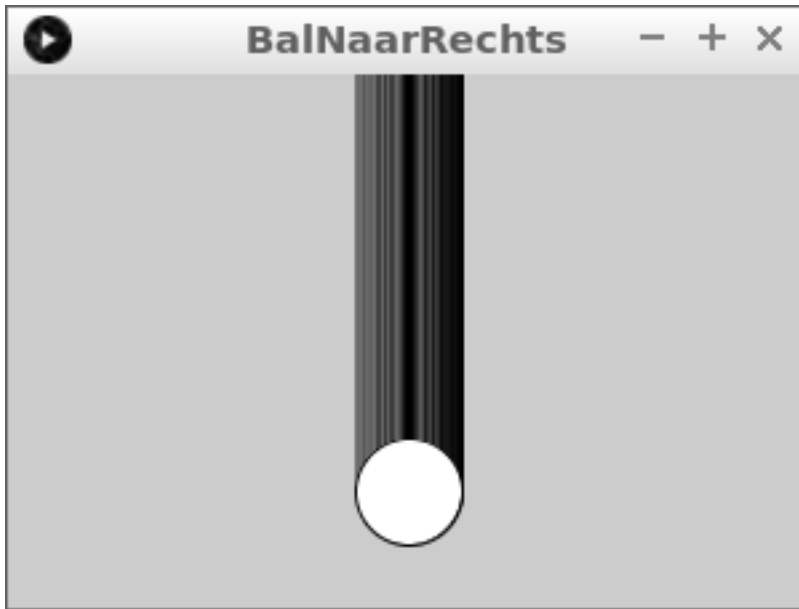


Figure 15: Bal naar rechts: opdracht 6

- Verander de naam van de variabele `x` in `y`
- Laat een bal aan de bovenkant van het scherm beginnen
- De bal moet 150 pixels naar rechts komen te staan
- De bal moet in een rechte lijn naar onder gaan. Tip: de bal staat nu op 50 pixels omlaag

## Bal naar rechts: oplossing 6

```
float y = 0;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(150, y, 40, 40);
  y = y + 1;
}
```

## Bal naar rechts: opdracht 7

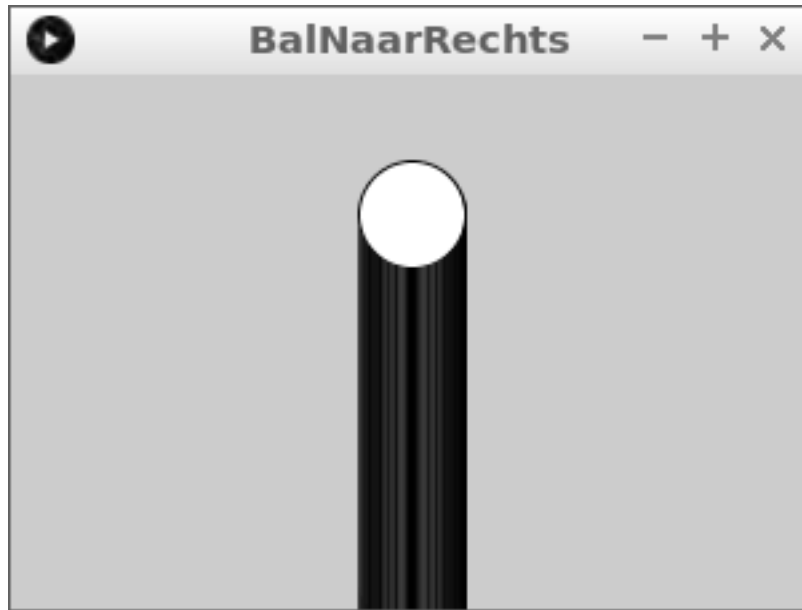


Figure 16: Bal naar rechts: opdracht 7

Nu gaan we de bal sneller en omhoog laten bewegen

- Laat een bal aan de onderkant van het scherm beginnen
- De bal moet in een rechte lijn naar boven gaan
- De bal moet twee keer zo snel gaan

## Bal naar rechts: oplossing 7

```
float y = 200;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(150, y, 40, 40);
  y = y - 1;
}
```



## Bal naar rechts: eindopdracht

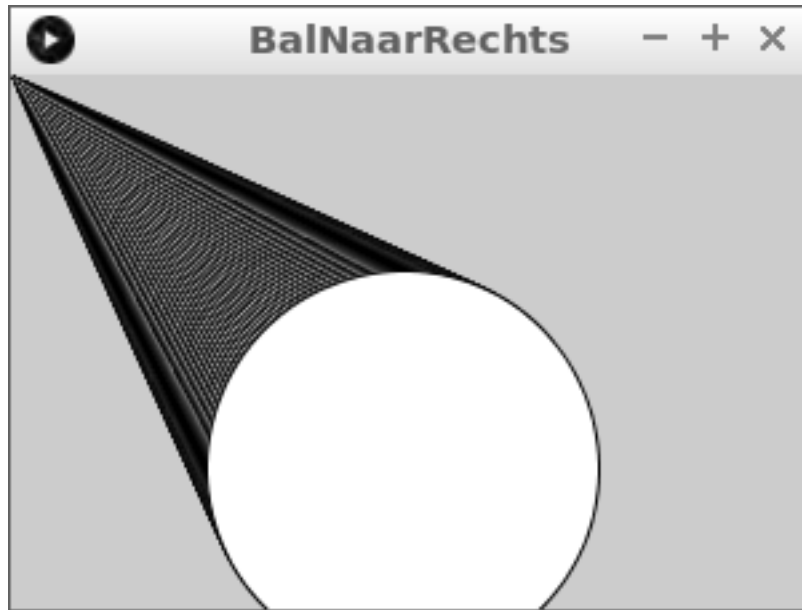


Figure 17: Bal naar rechts: eindopdracht

- de bal moet schuin naar rechts-omlaag gaan
- de bal moet groter worden in de breedte en hoogte
- zie ook figuur Eindopdracht 'Bal naar rechts'

## Bal naar rechts: links (naar websites)

- Eerste gedeelte: YouTube mp4
- Wat is een variabele: YouTube, mp4
- Tweede gedeelte: YouTube, mp4

## width en height

In deze les leer je hoe handig `width` en `height` zijn

### width en height: intro

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

void draw()
{
  ellipse(128, 128, 256, 256);
}
```



`size(800, 400);`

‘Lieve computer, maak een venster van 800 pixels breed en 400 pixels hoog.’

`ellipse(60,50,40,30);`

‘Lieve computer, teken een ovaal 60 pixels naar rechts, 50 pixels omlaag, die 40 pixels breed en 30 pixels hoog is.’

---

Type bovenstaande code over en run deze.

### width en height: opdracht 1

Maak het venster nu 128 bij 128 pixels klein.

### width en height: oplossing

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(64, 64, 128, 128);
}
```

## width en height

`width` en `height` zijn ingebouwd in Processing, zodat je programma nog werkt als je de grootte van je scherm aanpast.

Nu werken onze programma's alleen voor een scherm van een bepaalde grootte. Dan moet je elke keer als je een nieuwe grootte kiest, een heleboel code opnieuw typen!

Als we de breedte en hoogte van het scherm weten, weten we ook welke getallen in `ellipse` moeten:

- de x coördinaat van de ovaal is de helft van de breedte

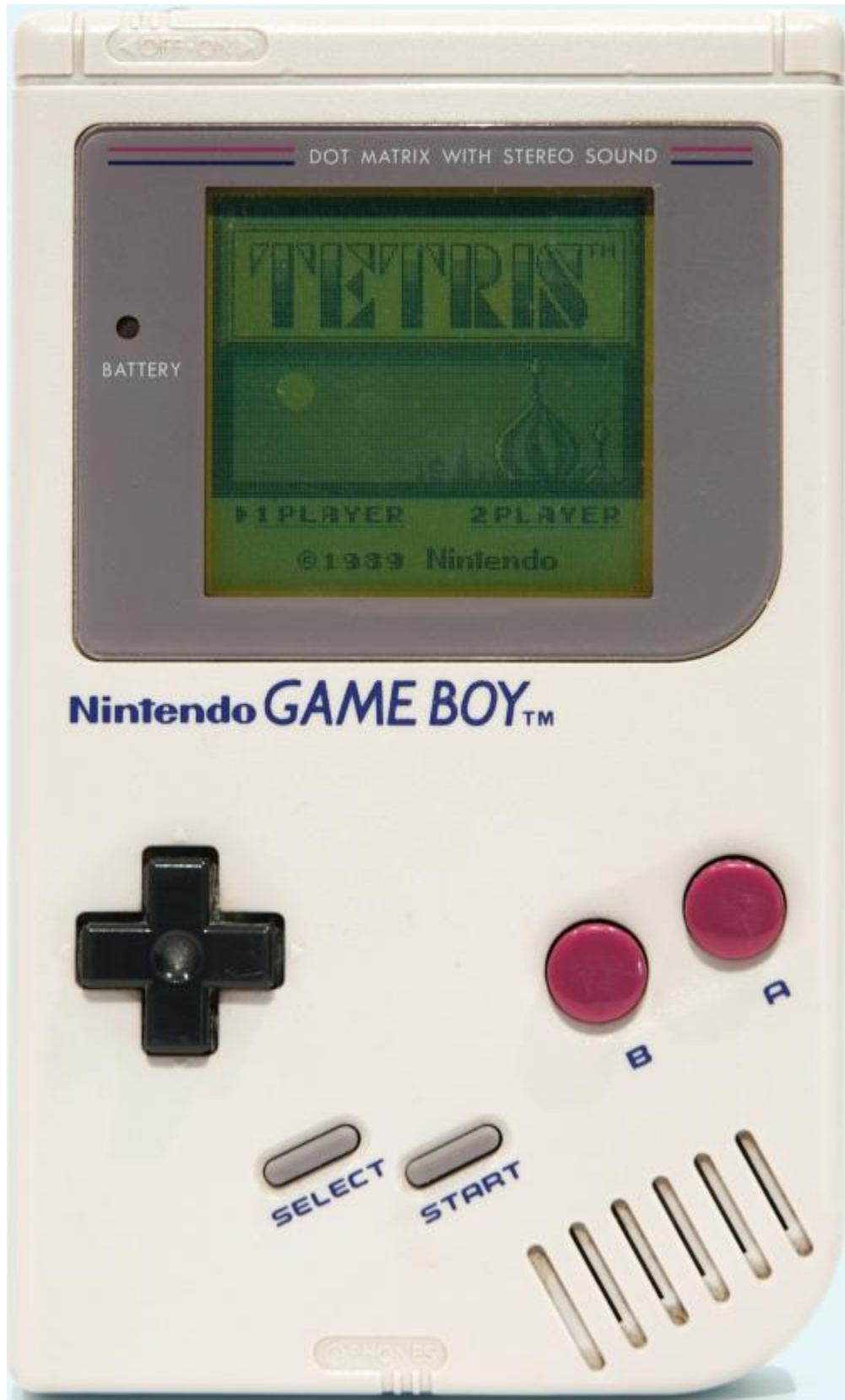


Figure 18: De Gameboy heeft een scherm van 160 bij 144 pixels

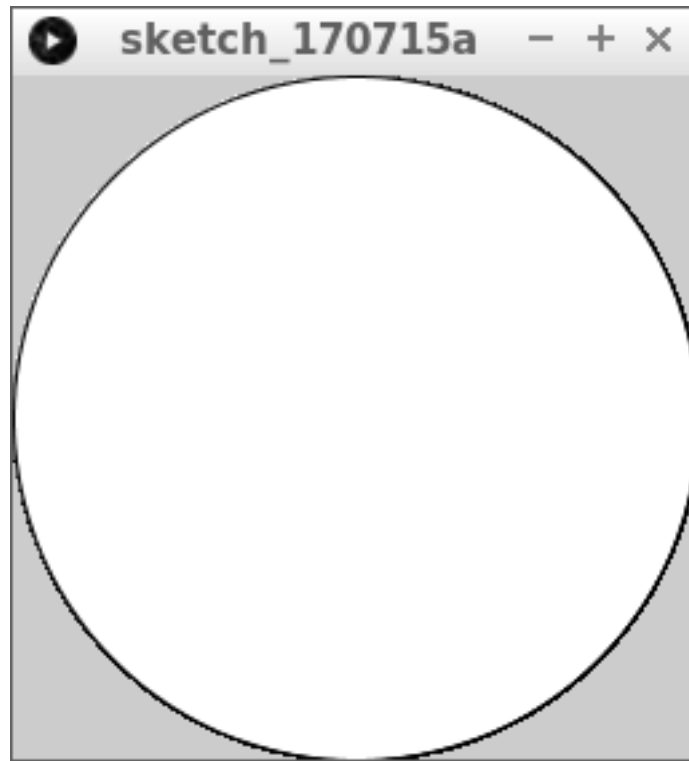


Figure 19: width en height: intro

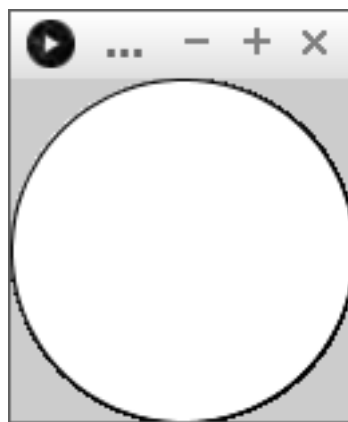


Figure 20: width en height: opdracht 1

- de y coördinaat van de ovaal is de helft van de hoogte
- de breedte van de ovaal is de breedte van het scherm
- de hoogte van de ovaal is de hoogte van het scherm

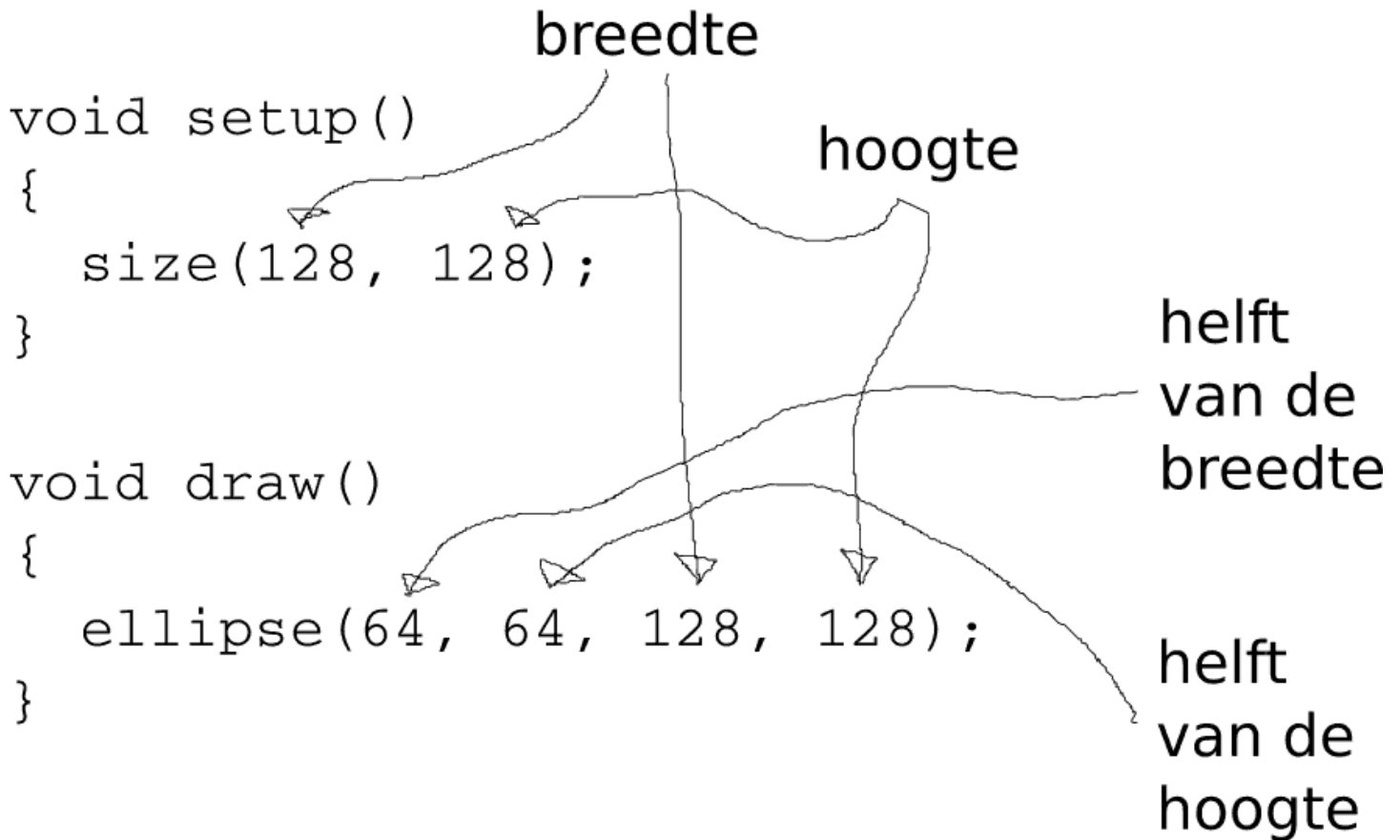




Figure 21: Wat je wilt zeggen

Processing weet de breedte en hoogte van het scherm: De breedte van het scherm heet **width** en de hoogte heet **height**

	
<b>width</b>	'Lieve computer, vul hier in hoeveel pixels het venster breed is.'
<b>height</b>	'Lieve computer, vul hier in hoeveel pixels het venster breed is.'

Deze getallen worden bepaald zodra je size gebruikt om de grootte van je scherm te definiëren.

## width en height: opdracht 2

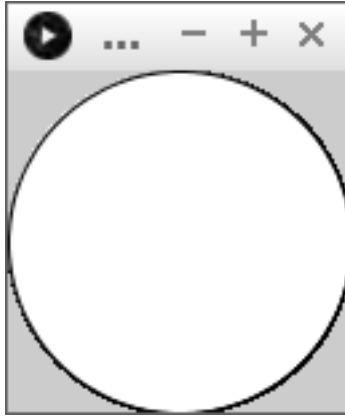


Figure 22: width en height: opdracht 2

Maak een programma wat een ovaal tekent die het scherm opvult:

- Verander de eerste 64 in `width / 2`
- Verander de tweede 64 in `height / 2`
- Verander de eerste 128 in `width`.
- Verander de tweede 128 in `height`.



---

/ 'gedeeld door', een deelstreep zoals je ook bij breuken hebt, :

---

## width en height: oplossing 2

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(width / 2, height / 2, width, height);
}
```

## width en height: opdracht 3

Zet het middelpunt van de cirkel op coördinaat (0, 0).

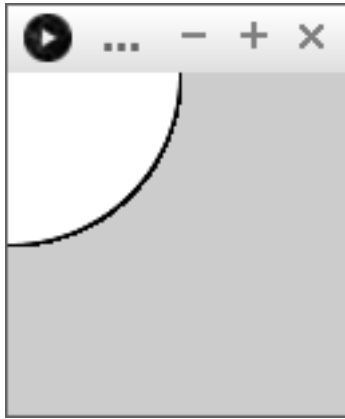


Figure 23: width en height: opdracht 3

### width en height: oplossing 3

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
}
```

### width en height: opdracht 4

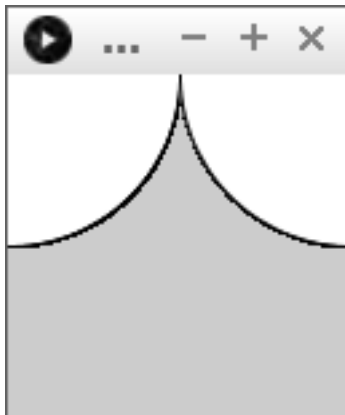


Figure 24: width en height: opdracht 4

Maak een tweede cirkel die als middelpunt de rechterbovenhoek heeft. Gebruik `width` en/of `height`.

#### width en height: oplossing 4

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
  ellipse(width, 0, width, height);
}
```

#### width en height: opdracht 5

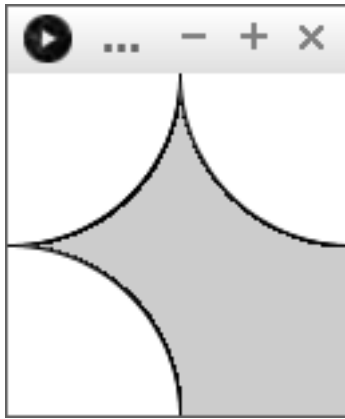


Figure 25: width en height: opdracht 5

Maak een derde cirkel die als middelpunt de linkeronderhoek heeft. Gebruik `width` en/of `height`.

#### width en height: oplossing 5

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
  ellipse(width, 0, width, height);
  ellipse(0, height, width, height);
}
```



**width en height: eindopdracht**

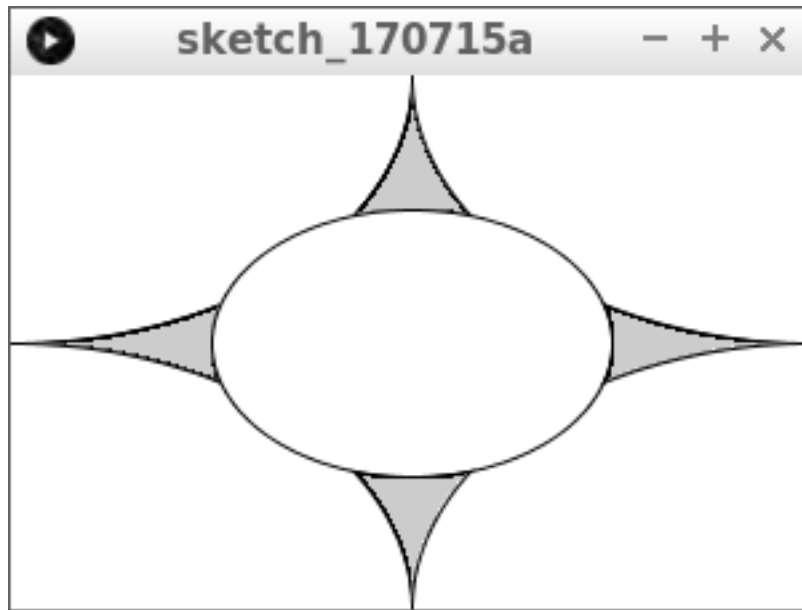


Figure 26: **width en height:** eindopdracht

- Maak het venster 300 pixels breed en 200 pixels hoog
- Maak een vierde cirkel die als middelpunt de rechteronderhoek heeft
- Maak een vijfde cirkel die in het midden staat en twee keer zo klein is

**width en height: links**

- **width en height:** YouTube, mp4

## point en random

In deze les gaan we leren

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



Figure 27: Eindopdracht

## point en random: intro

Pixels zijn de vierkantjes waaruit je beeldscherm is opgebouwd.



Pixel = een vierkantje op je beeldscherm

---

Hoe meer pixels je scherm heeft, hoe scherper het beeld eruit ziet. Dat zie je goed bij oude games: die hebben minder pixels:

## point en random: opdracht 1

Run de volgende code:

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(150, 100);
}
```



Figure 28: Super Mario Bros 1



```
point(150, 100);
```

‘Lieve computer, teken een puntje op de pixel die tweehonderd pixels naar rechts en honderdvijftig pixels omlaag is’

```
point(150, 100);
```

‘Lieve computer, teken een puntje op coördinaat (150, 100)’

point en random: oplossing 1



Figure 29: point en random: oplossing 1

## point en random: opdracht 2

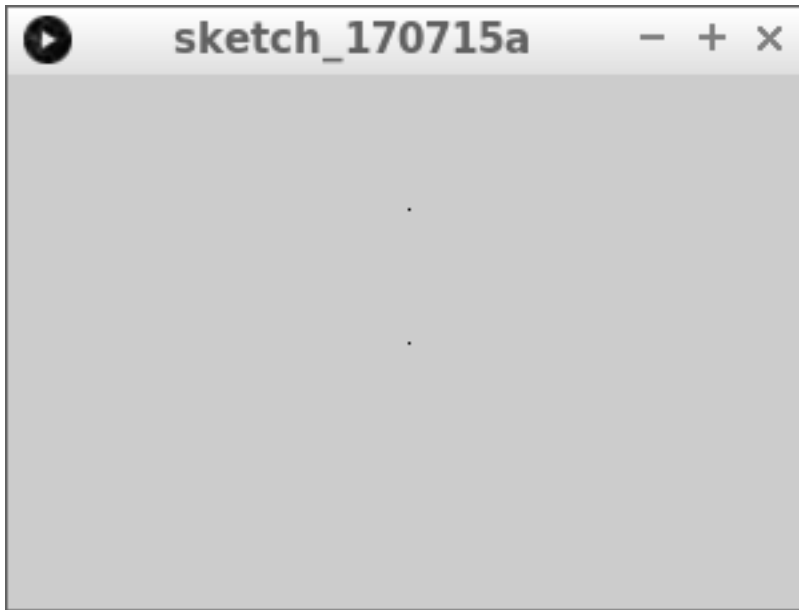


Figure 30: point en random: opdracht 2

Teken een tweede puntje tussen de eerste en de bovenkant van het venster.

## point en random: oplossing 2

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(150, 100);
  point(150, 50);
}
```

## point en random: opdracht 3

De eerste pixel zit precies in het midden. Oftewel op de helft van de breedte van het venster en op de helft van de hoogte van het scherm. Verander `point(150,100);` naar iets met `width` en `height`.

## point en random: oplossing 3

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(width / 2, height / 2);
  point(150, 50);
}
```



---

`width / 2`

‘Lieve computer, vul hier de breedte van het venster in, gedeeld door twee’

---

### point en random: opdracht 4

De tweede pixel zit

- op de helft van de breedte van het venster
- op een kwart van de hoogte van het scherm

Verander `point(150, 50)`; naar iets met `width` en `height`.

### point en random: oplossing 4

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
}
```



---

`height / 4`

‘Lieve computer, vul hier de hoogte van het venster in, gedeeld door vier’

---

### point en random: opdracht 5

Teken een nieuwe pixel, in de linkerbovenhoek van het scherm.

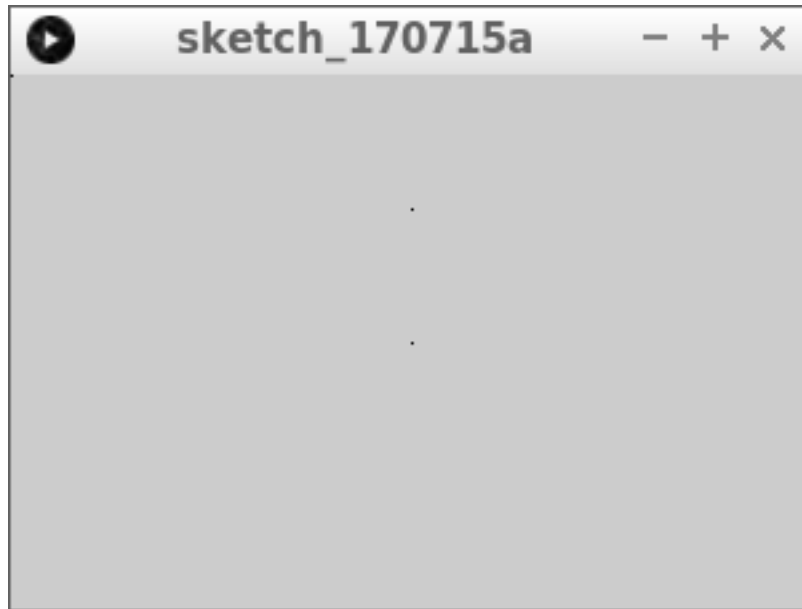


Figure 31: point en random: opdracht 5

### point en random: oplossing 5

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
}
```



---

point(0,0);	‘Lieve computer, teken een puntje in de linkerbovenhoek’
point(0,0);	‘Lieve computer, teken een puntje op coördinaat (0, 0)’

---

### point en random: opdracht 6

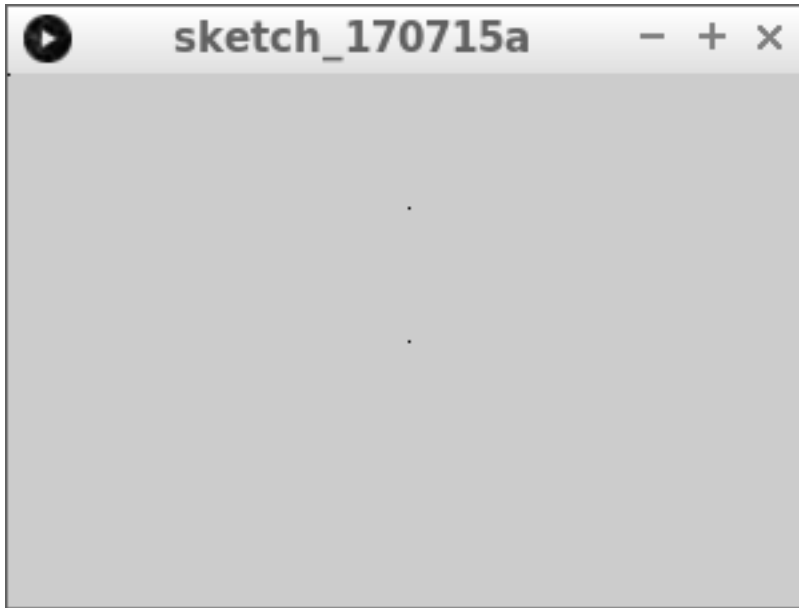


Figure 32: point en random: opdracht 6

Teken een nieuwe pixel, in de rechtbovenhoek van het scherm. Gebruik `width - 1` als eerste getal binnen de ronde haakjes van `point`.

### point en random: oplossing 6

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
  point(width - 1, 0);
}
```

### point en random: opdracht 7

Teken twee pixels erbij, in de onderste twee hoeken. Gebruik `width - 1` en `height - 1` op de juiste plekken.



Figure 33: point en random: opdracht 7

### point en random: oplossing 7

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
  point(width - 1, 0);
  point(0, height - 1);
  point(width - 1, height - 1);
}
```

### point en random: opdracht 8

Run deze code:

```
void setup()
{
  size(300, 200);
}

void draw()
{
  point(random(300), 100);
}
```

Wat zie je?



## point en random: oplossing 8

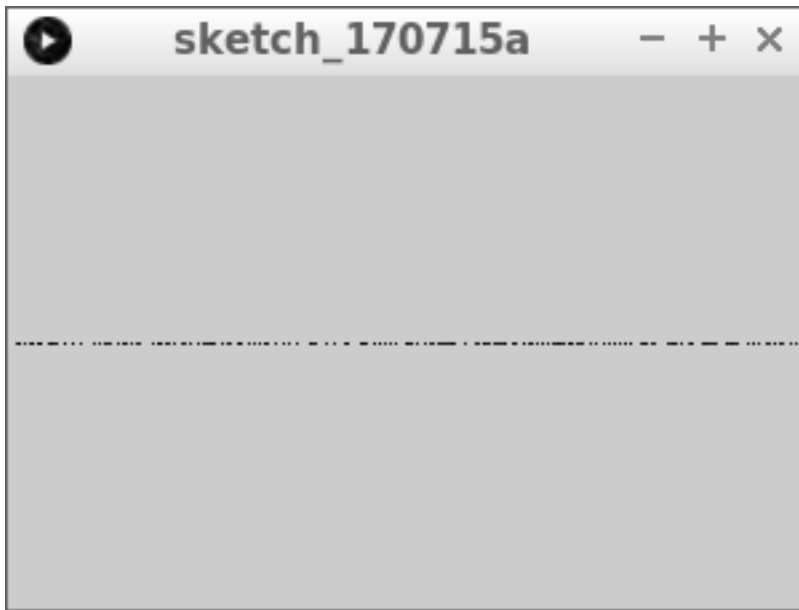


Figure 34: point en random: oplossing 8

Je ziet dat er puntjes op willekeurige plekken worden getekend, maar wel altijd op dezelfde hoogte.



## point en random: opdracht 9



Figure 35: point en random: opdracht 9

Maak het venster 400 pixels breed en 100 pixels hoog. Gebruik in plaats van `random(300)` iets met `random` en `width`. Zorg dat de lijn van puntjes op de halve hoogte van het scherm blijft.

## point en random: oplossing 9

```
void setup()
{
  size(400, 100);
}

void draw()
{
  point(random(width), height / 2);
}
```

## point en random: eindopdracht



Figure 36: Eindopdracht

Laat de computer willekeurig puntjes tekenen in het hele venster.

## Links

- YouTube
- mp4