



Figure 1: Boek 1

Contents

Voorwoord	1
Een Mooi Programma	2
Bal naar rechts	5
width en height	16
point en random	23

Voorwoord



Figure 1: Het logo van De Jonge Onderzoekers



Figure 2: Het logo van Codestarter

Dit is het Processing boek van de Dojo. Processing is een programmeertaal. Dit boek leert je die programmeertaal.

Over dit boek

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



Figure 3: De licentie van dit boek

(C) Dojo Groningen 2016-2017

Het is nog een beetje een slordig boek. Er zitten tiepvauten in en de opmaak is niet *altijd even mooi*.

Daarom staat dit boek op een GitHub. Om precies te zijn, op <https://github.com/richelbilderbeek/Dojo>. Hierdoor 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:

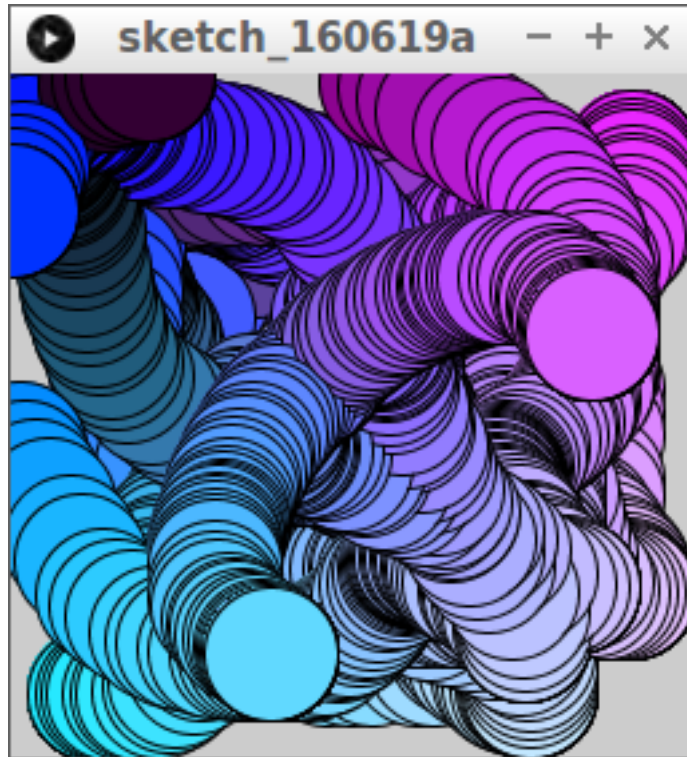


Figure 4: EenMooiProgramma

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:

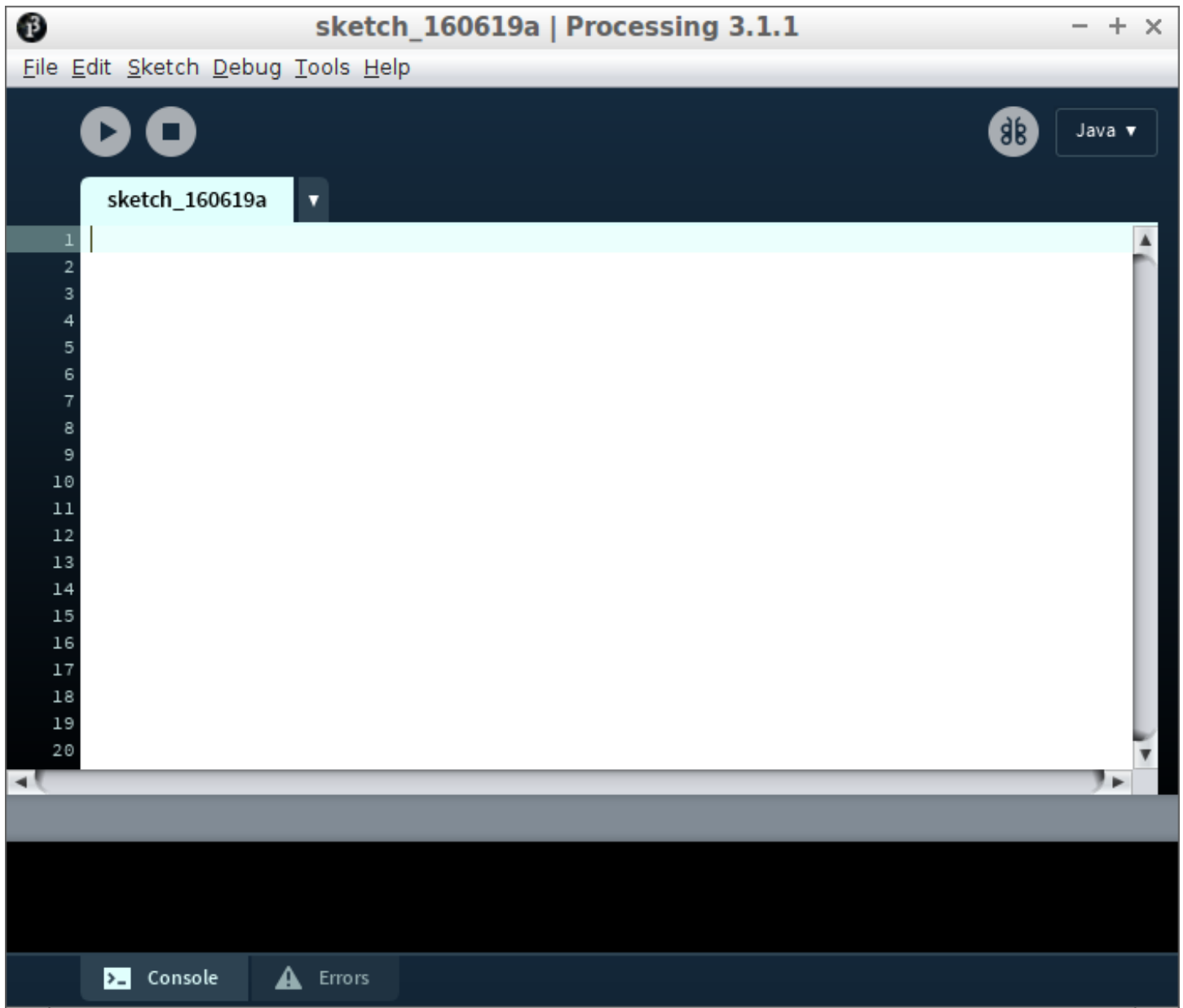


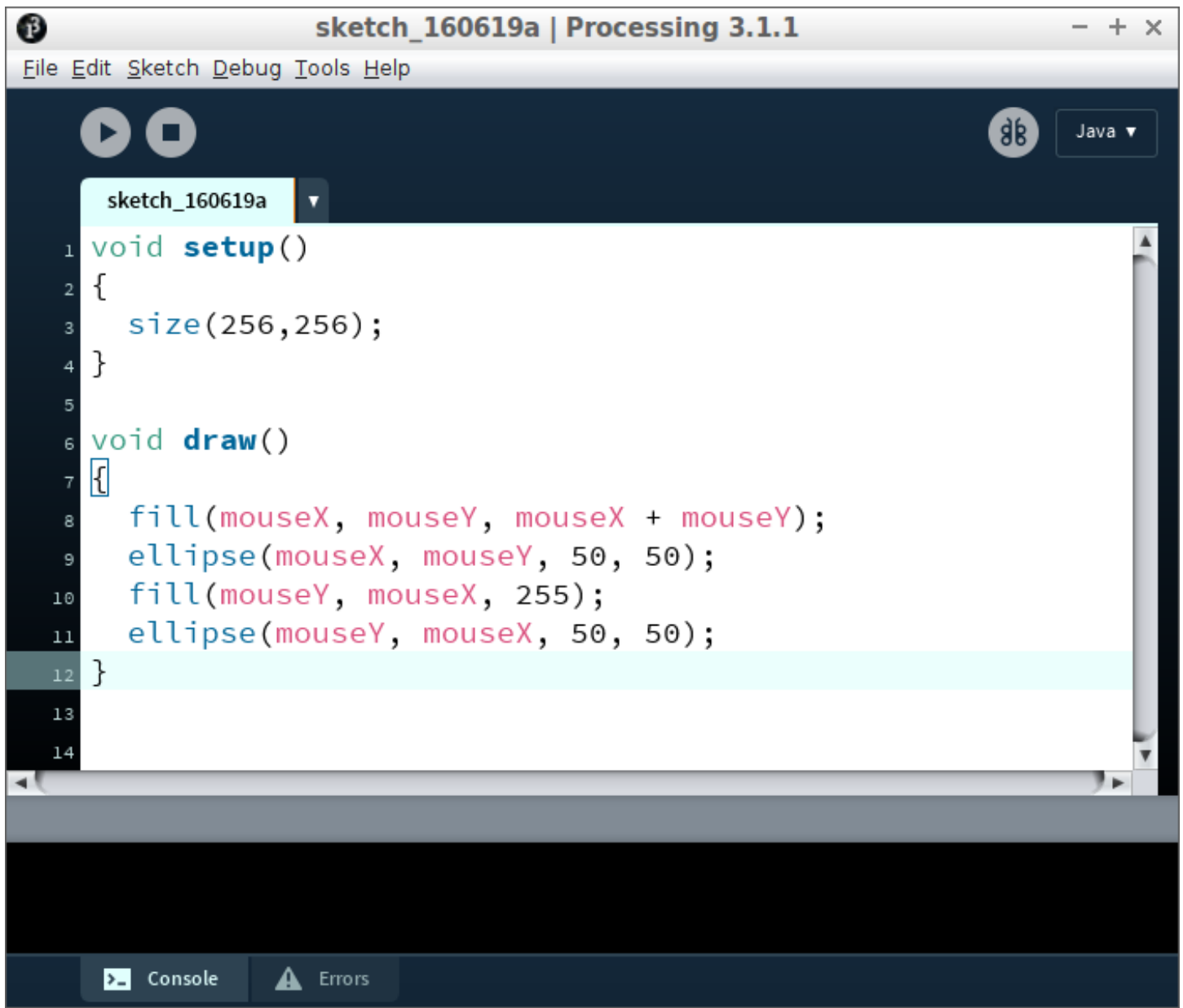
Figure 5: Processing zonder code

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.

The image shows a screenshot of the Processing IDE window. The title bar reads 'sketch_160619a | Processing 3.1.1'. The menu bar includes 'File', 'Edit', 'Sketch', 'Debug', 'Tools', and 'Help'. Below the menu bar are buttons for 'Run' (a play icon) and 'Stop' (a square icon), and a language dropdown menu set to 'Java'. The main text area contains the following code:

```
1 void setup()  
2 {  
3   size(256,256);  
4 }  
5  
6 void draw()  
7 {  
8   fill(mouseX, mouseY, mouseX + mouseY);  
9   ellipse(mouseX, mouseY, 50, 50);  
10  fill(mouseY, mouseX, 255);  
11  ellipse(mouseY, mouseX, 50, 50);  
12 }  
13  
14
```

The code is color-coded: keywords like 'void', 'setup', 'draw', 'size', 'fill', and 'ellipse' are in blue; variables like 'mouseX' and 'mouseY' are in pink; and numbers and punctuation are in black. The bottom of the window has tabs for 'Console' and 'Errors'.

Figure 6: Processing met code

Eindopdracht

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



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

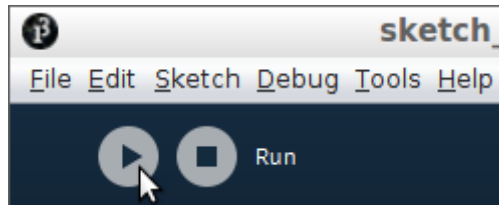


Figure 7: De Run knop

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.

Bal naar rechts

Type de volgende code over:

```
float x = 60;

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

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 Een bal die naar rechts beweegt).

Opdracht 1

Het scherm is nu 600 pixels breed. Kun je deze 800 pixels breed krijgen? Verander de code en druk op 'Run'.

Oplossing 1

Er zit een 600 in de code. Deze naar 800 veranderen is genoeg:

```
float x = 60;

void setup()
{
  size(800, 400);
}

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

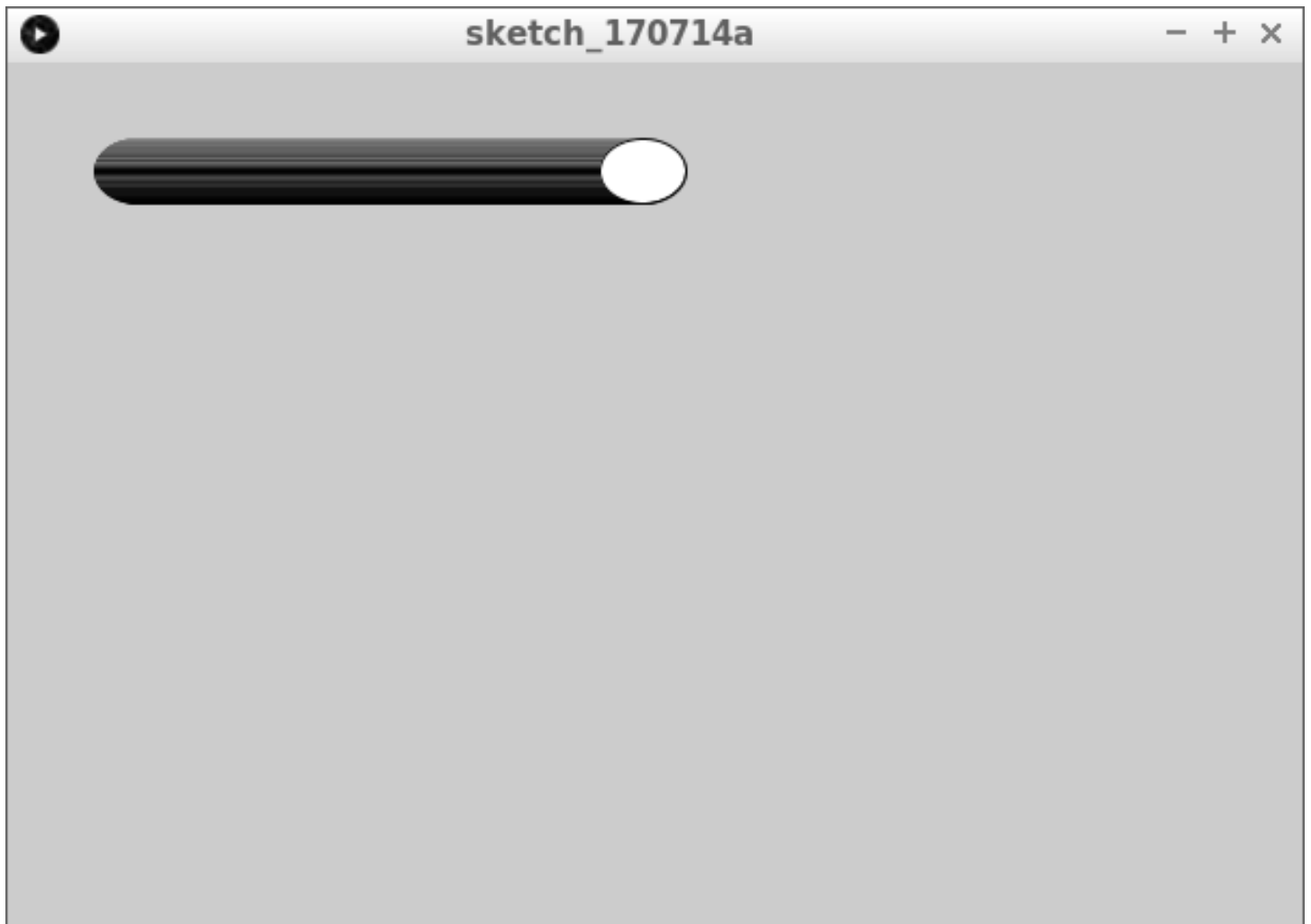


Figure 8: Een bal die naar rechts beweegt

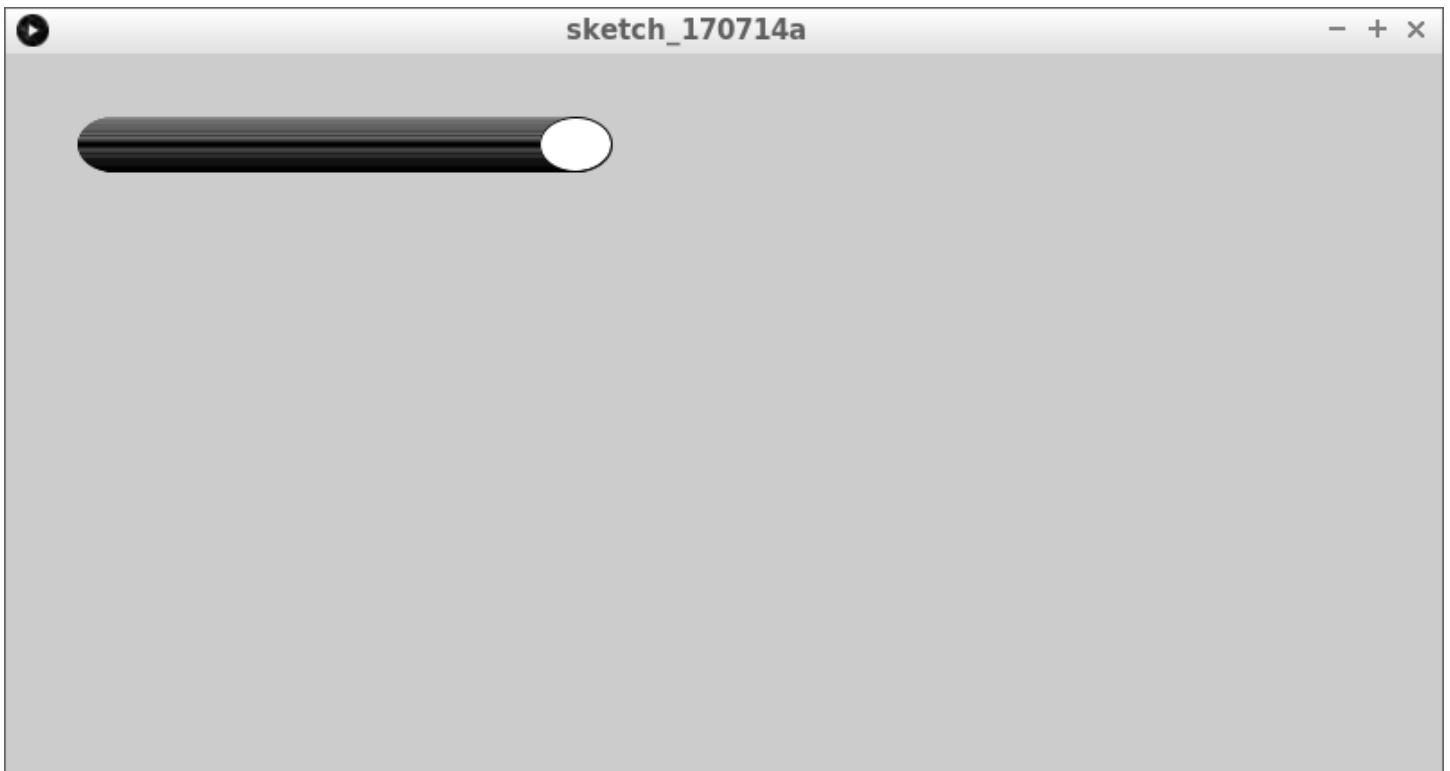




Figure 9: Opdracht 1

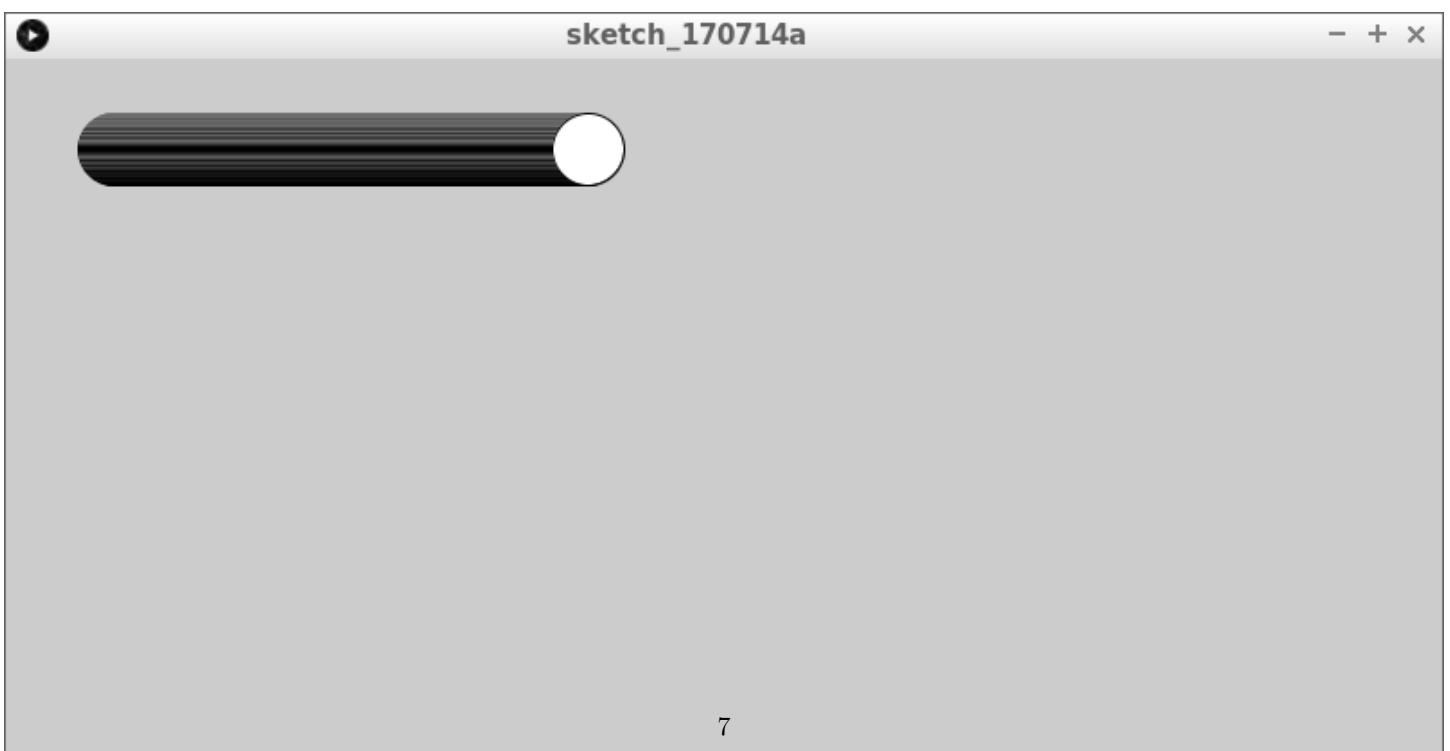




`size(800, 400);`

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

Opdracht 2





```
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.’

Opdracht 3

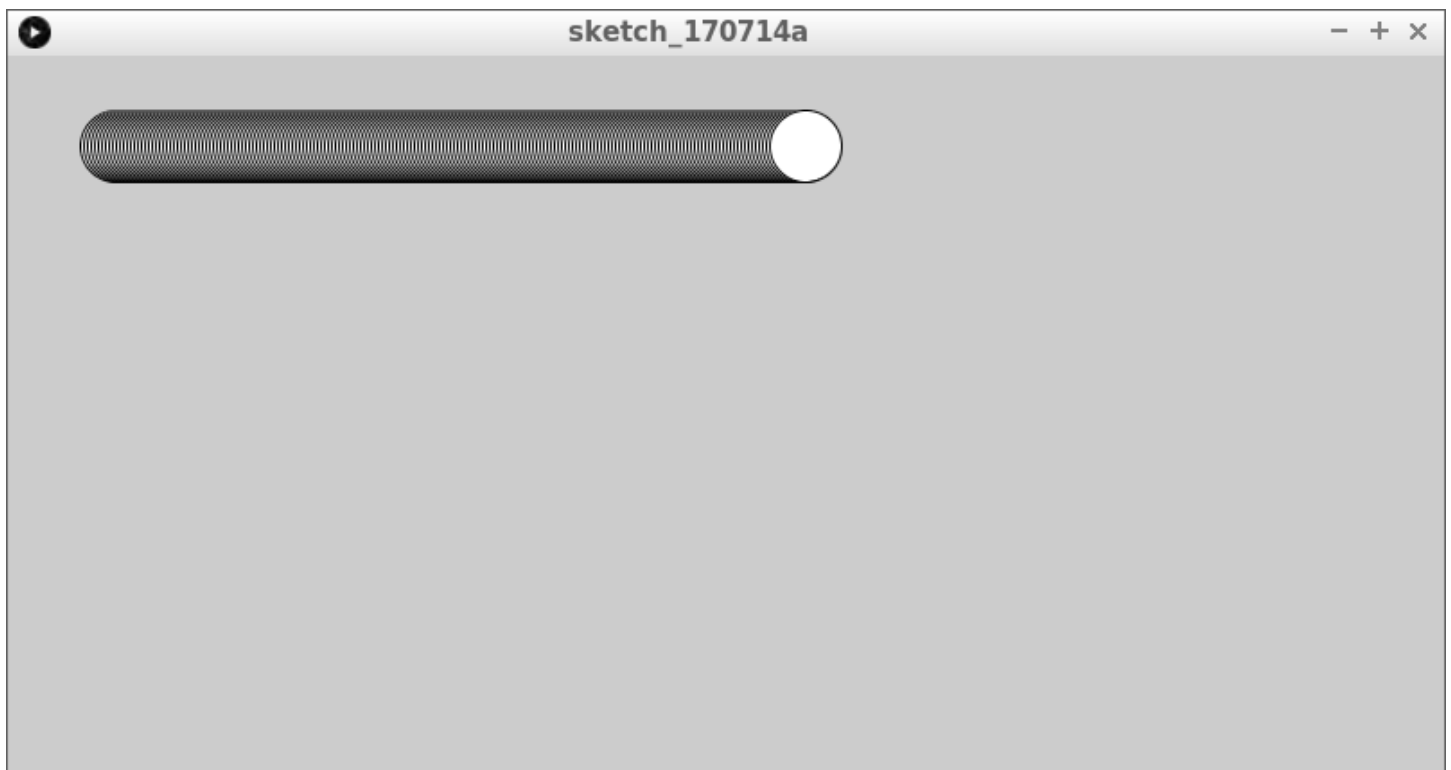


Figure 11: Opdracht 3

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



Oplossing 3

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

```
float x = 60;

void setup()
{
  size(800, 400);
}

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

	
---	---

<code>x = x + 1;</code>	'Lieve computer, maak x een hoger.'
<code>x += 1;</code>	'Lieve computer, maak x een hoger.'
<code>x++;</code>	'Lieve computer, maak x een hoger.'
<code>++x;</code>	'Lieve computer, maak x een hoger.'

Opdracht 4

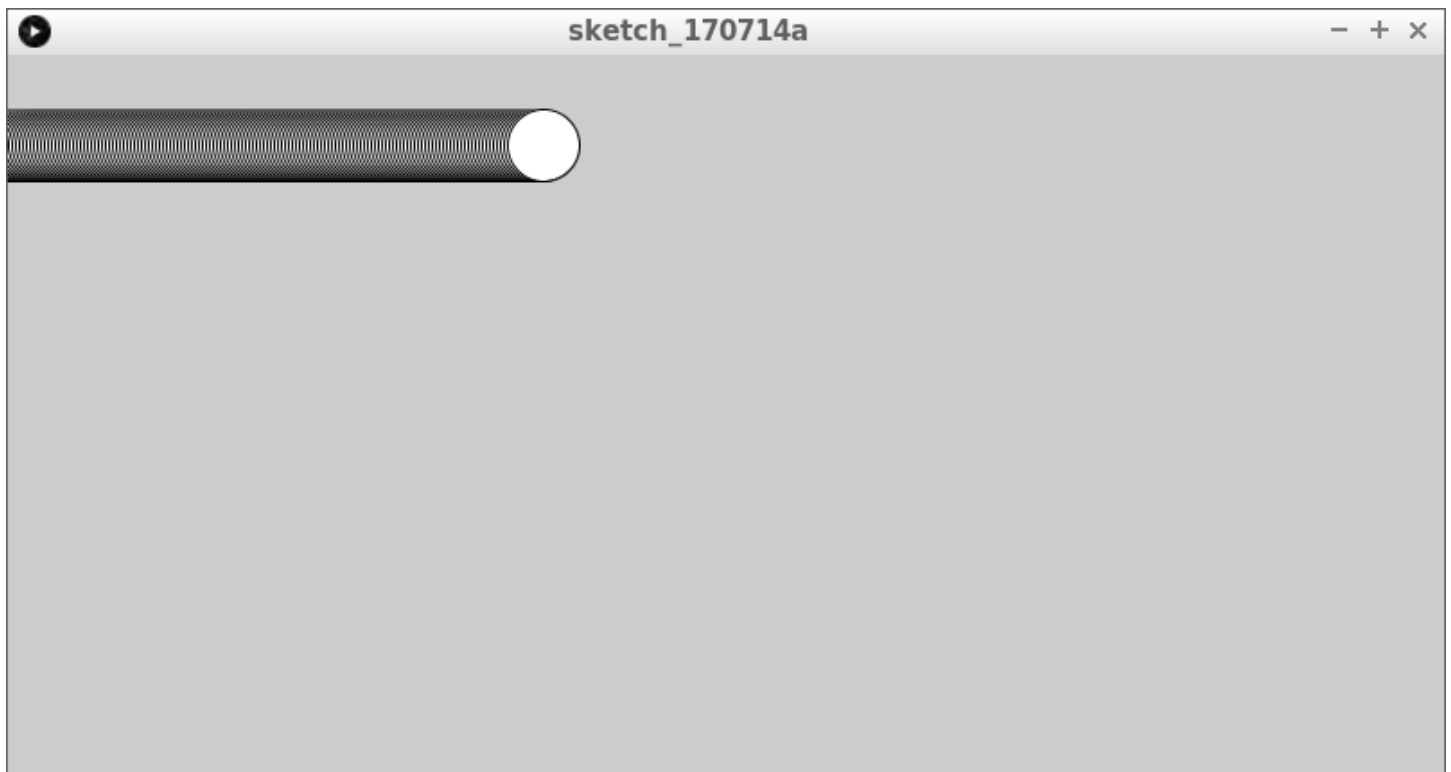


Figure 12: Opdracht 4

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

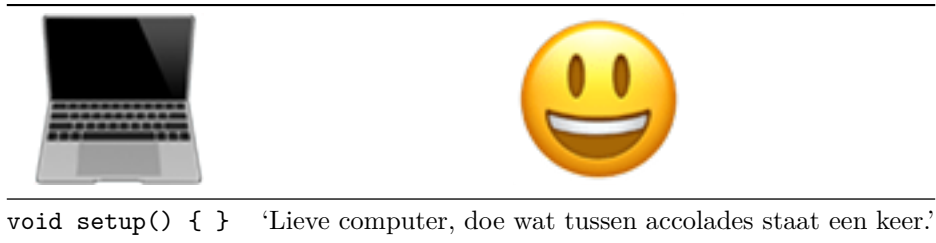
Oplossing 4

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

```
float x = 0;

void setup()
{
  size(800, 400);
}

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



Bal naar links

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

Opdracht 5

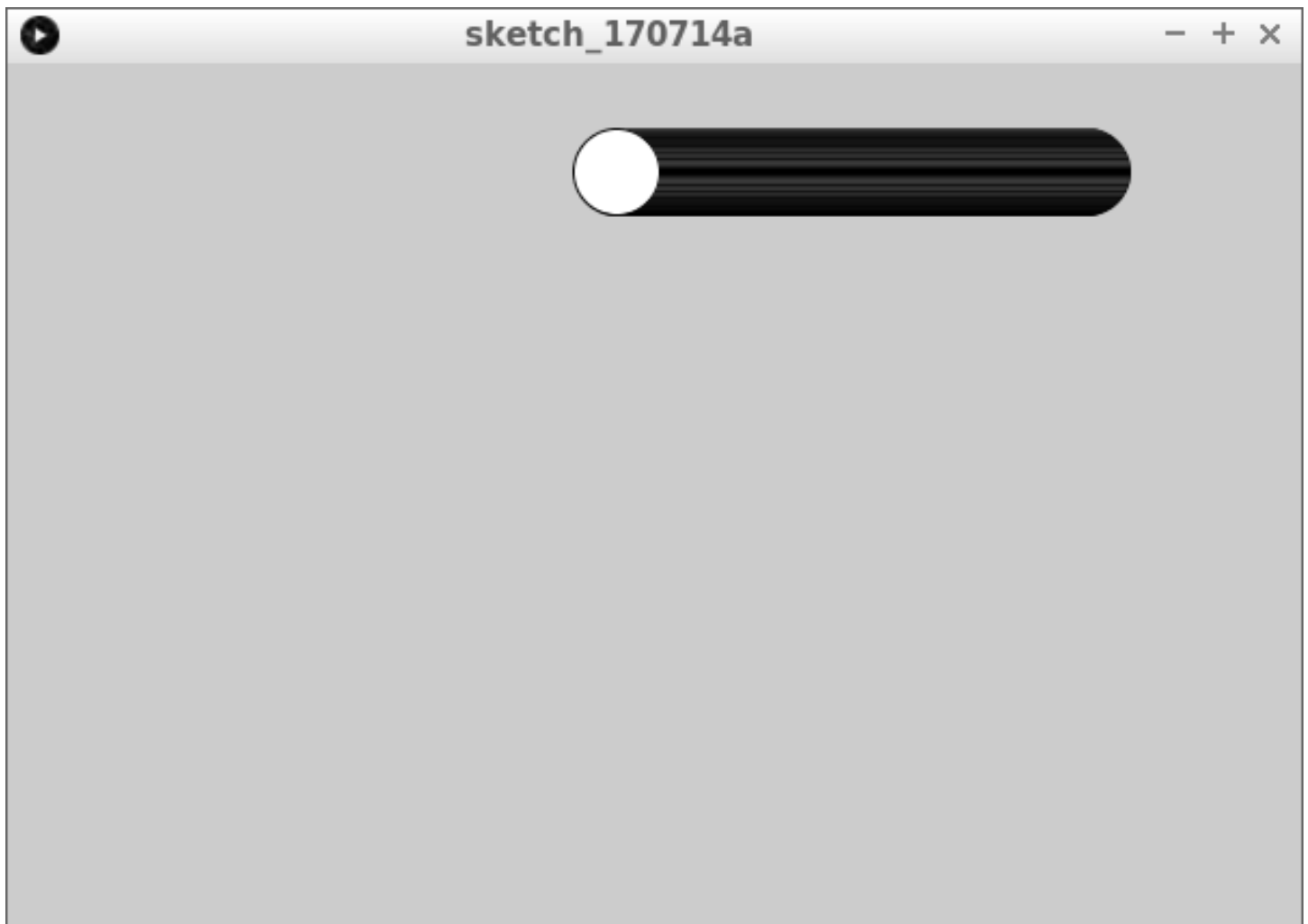


Figure 13: Opdracht 5

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

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 = 500;

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

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



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

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

Wat is een variabele?

In de eerste regel gebruiken we een variable:

```
float x = 50;
```

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



```
float x = 50;
```

‘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. Dit lijkt een beetje zoals jouw achternaam in het telefoonboek staat.

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.



```
geld
1000000
```

‘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

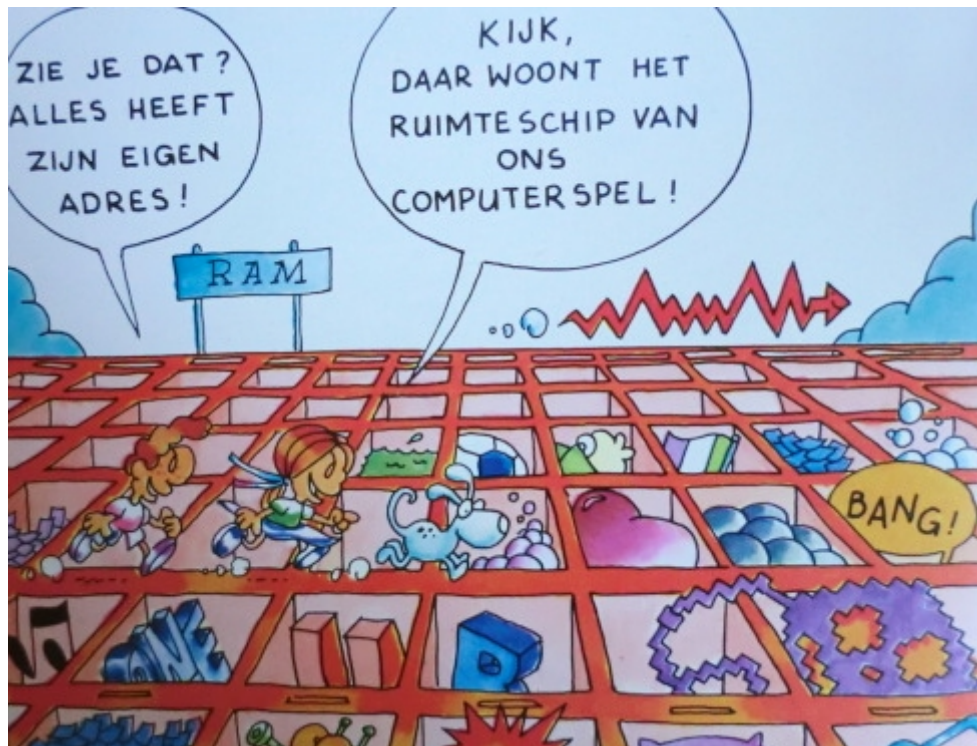




Figure 14: Het geheugen van een computer

het einde van een zin aan (zoals de punt in een Nederlandse tekst).

	
float	'Een komma getal'
=	'is vanaf nu'
;	'.'

Bal naar onder

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

Opdracht 6

- Verander de naam van de variabele `x` in `y`
- Laat een bal aan de bovenkant van het scherm beginnen
- De bal moet 60 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

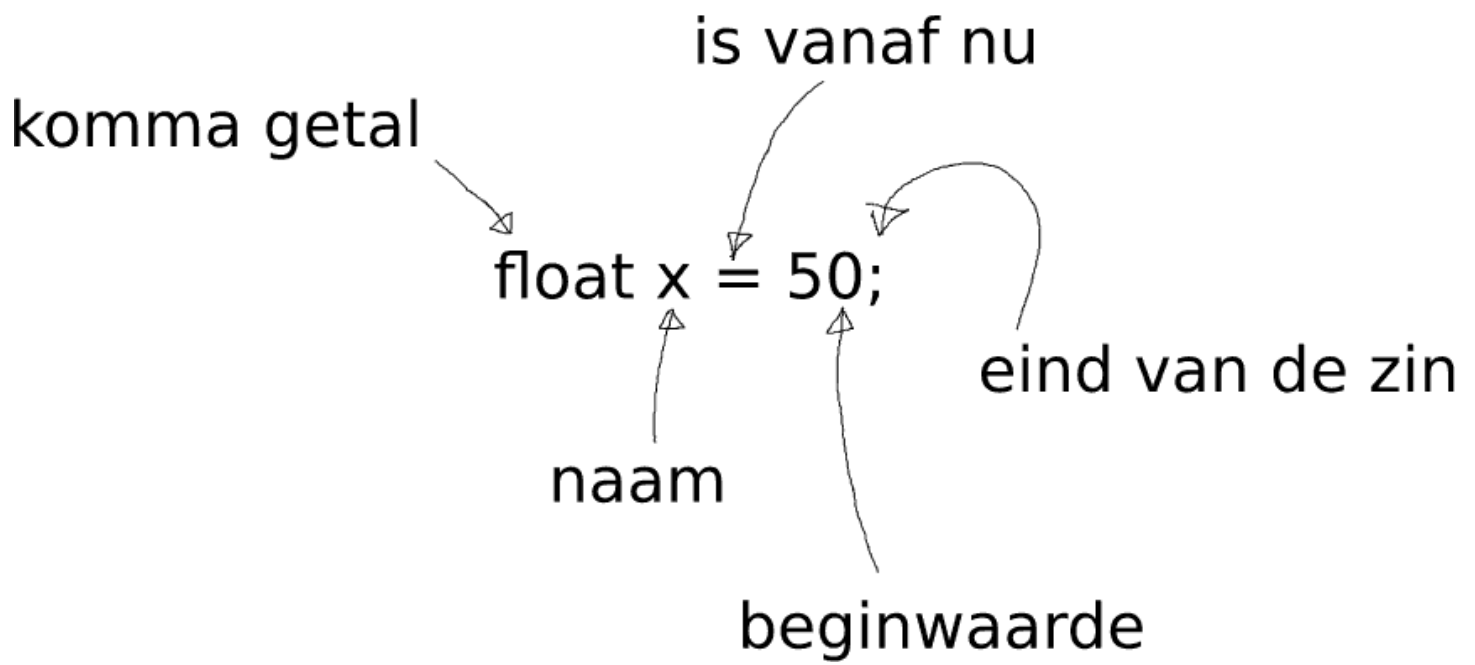


Figure 15: Uitleg van `float x = 50;`



Figure 16: Opdracht 6

Oplossing 6

```
float y = 50;

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

void draw()
{
  ellipse(60,y,40,40);
  y = y + 1;
}
```

Opdracht 7

Nu gaan we de bal sneller en omhoog laten bewegen



Figure 17: Opdracht 7

- 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

Oplossing 7

```
float y = 300;

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

void draw()
{
  ellipse(50, y, 50, 50);
  y = y - 2;
}
```

Eindopdracht

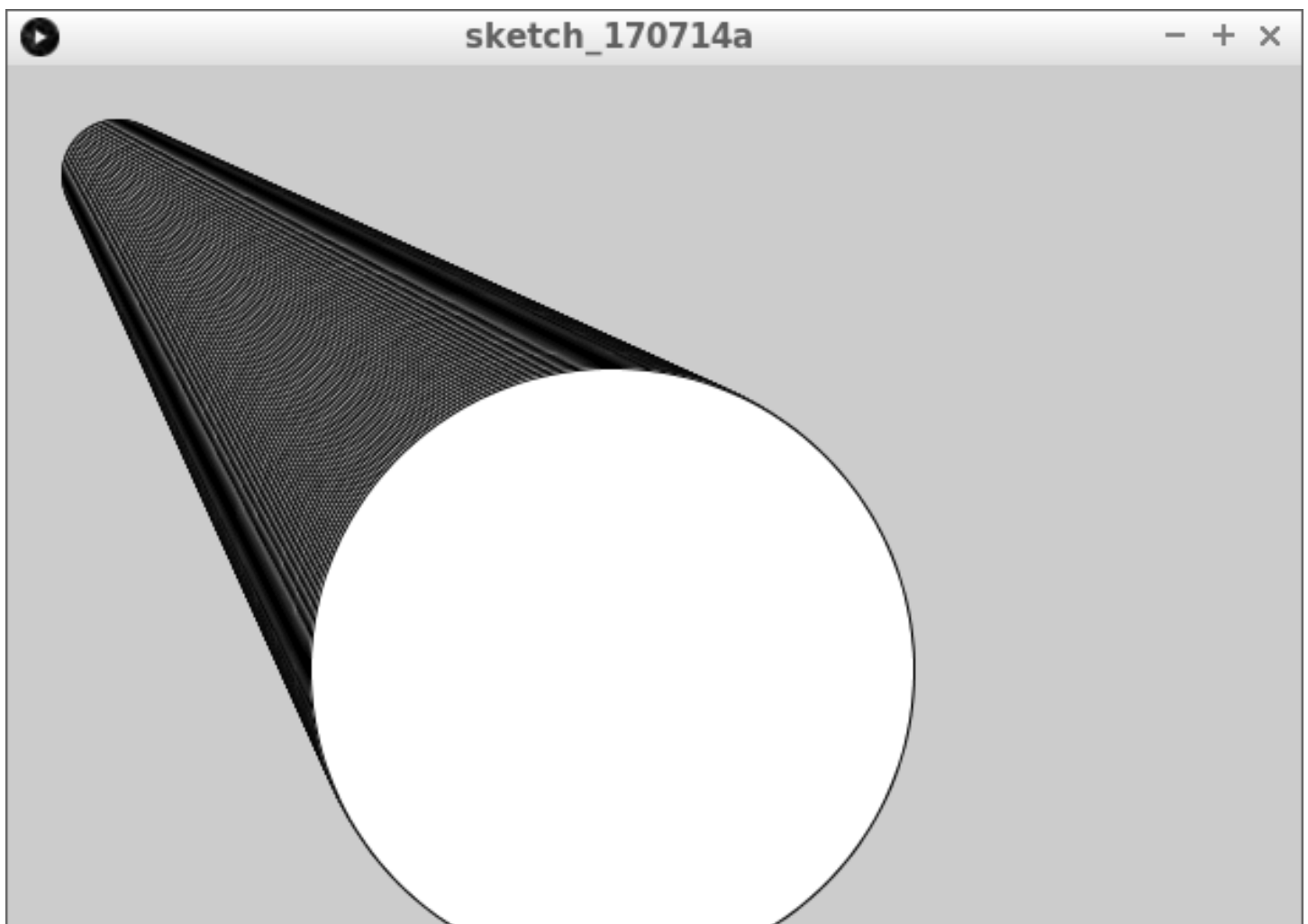


Figure 18: Eindopdracht 'Bal naar rechts'

- gebruik een variabele t (van tijd)
- 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'

width en height

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

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.’

Opdracht 1

Type bovenstaande code over en run deze.

Oplossing 1

Opdracht 2

Maak het venster nu 128 bij 128 pixels klein.

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.

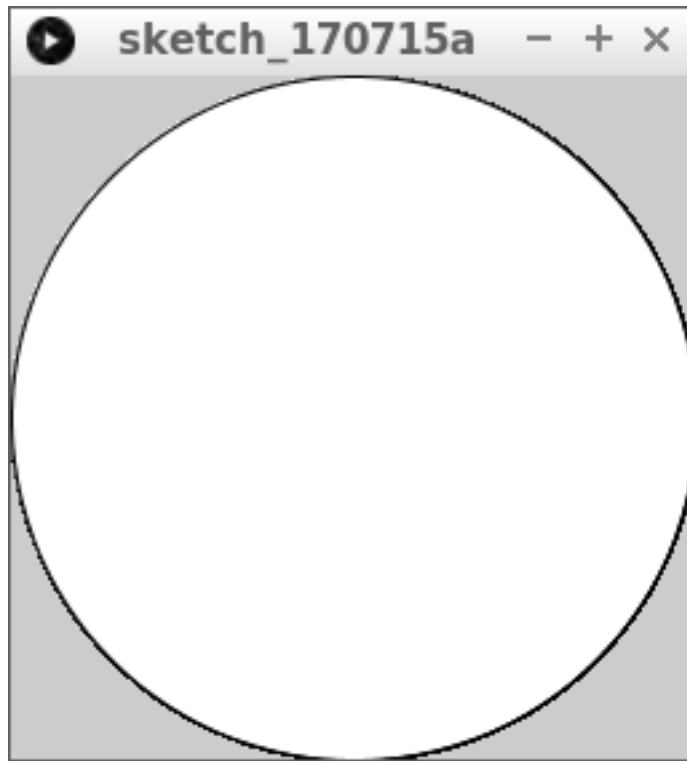


Figure 19: Oplossing 1

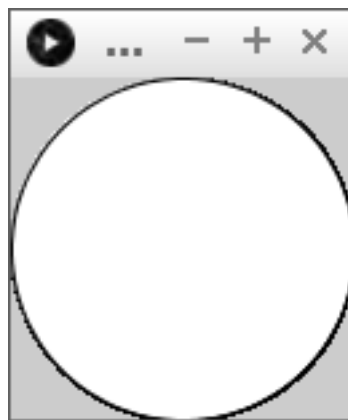


Figure 20: Opdracht 2

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
- 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

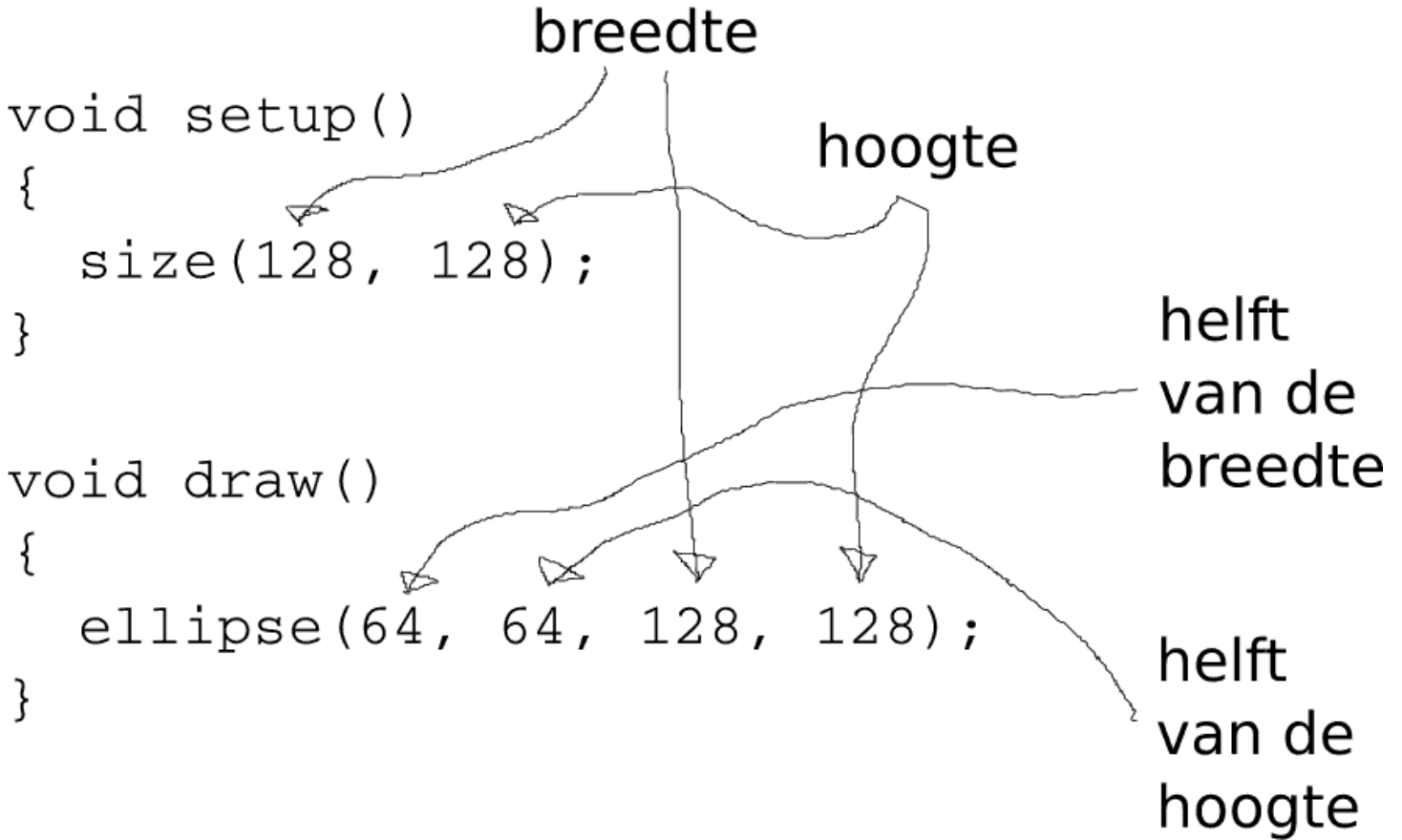




Figure 21: Wat je wilt zeggen

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

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

Opdracht 3

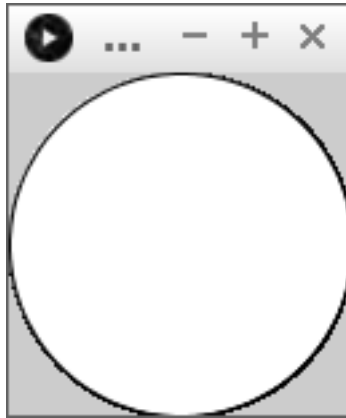


Figure 22: Opdracht 3

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, :

Oplossing 3

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

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

Opdracht 4

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

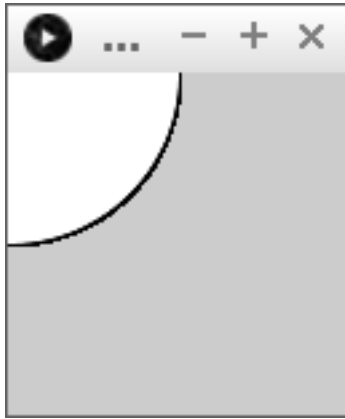


Figure 23: Opdracht 4

Oplossing 4

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

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

Opdracht 5

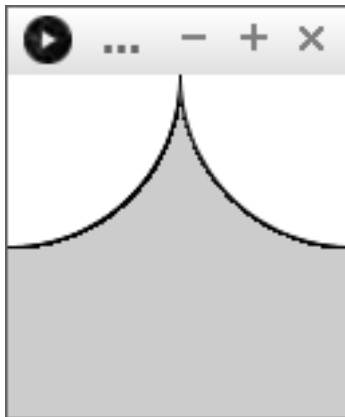


Figure 24: Opdracht 5

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

Oplossing 5

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

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

Opdracht 6

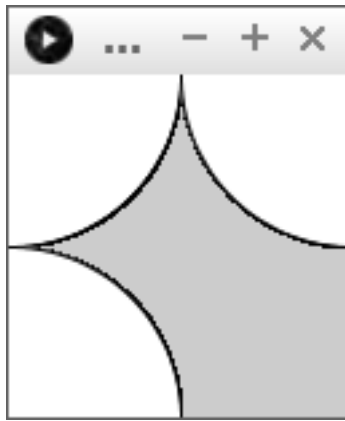


Figure 25: Opdracht 6

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

Oplossing 6

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

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

Eindopdracht

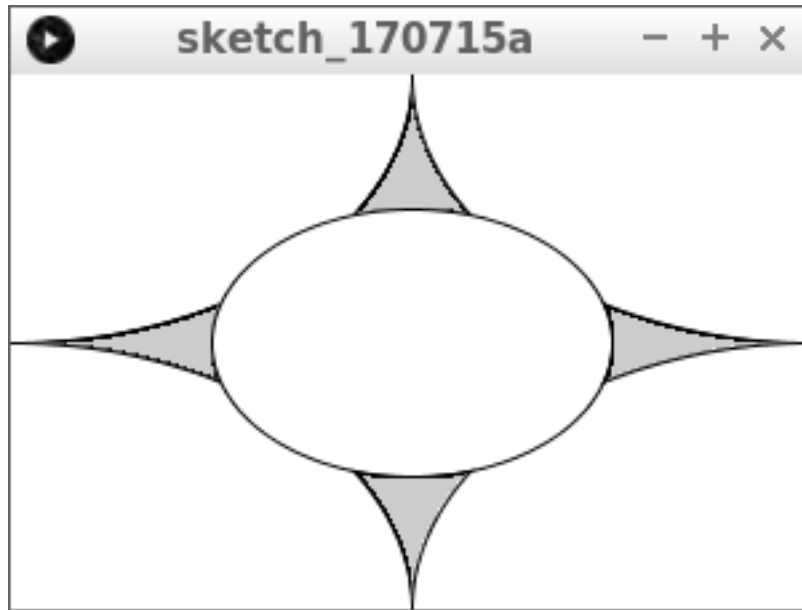


Figure 26: Eindopdracht 'width' en 'height'

- 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

point en random

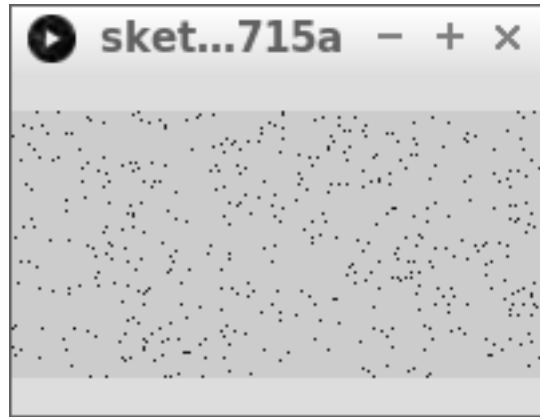


Figure 27: Eindopdracht

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

Pixels

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:

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)’

Oplossing 1



Figure 29: Oplossing 1

Opdracht 2

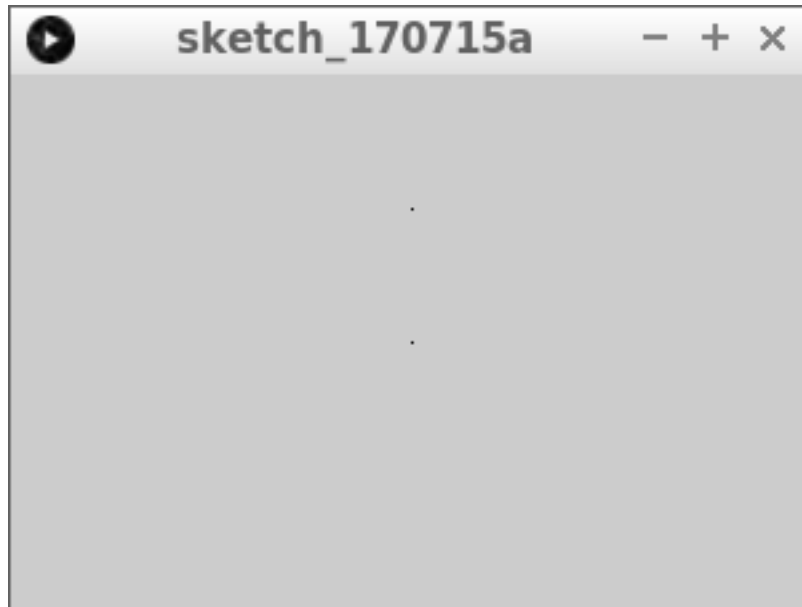


Figure 30: Opdracht 2

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

Oplossing 2

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

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

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`.

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’

Opdracht 3.14

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`.

Oplossing 3.14

```
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’

Opdracht 4

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

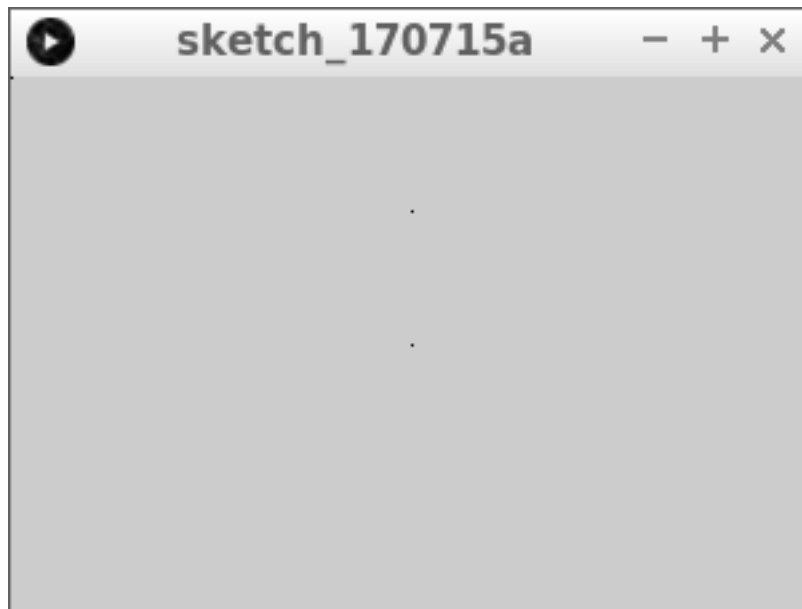


Figure 31: Opdracht 4

Oplossing 4

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

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



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

Opdracht 5



Figure 32: Opdracht 5

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

Oplossing 5

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

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

Opdracht 6

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

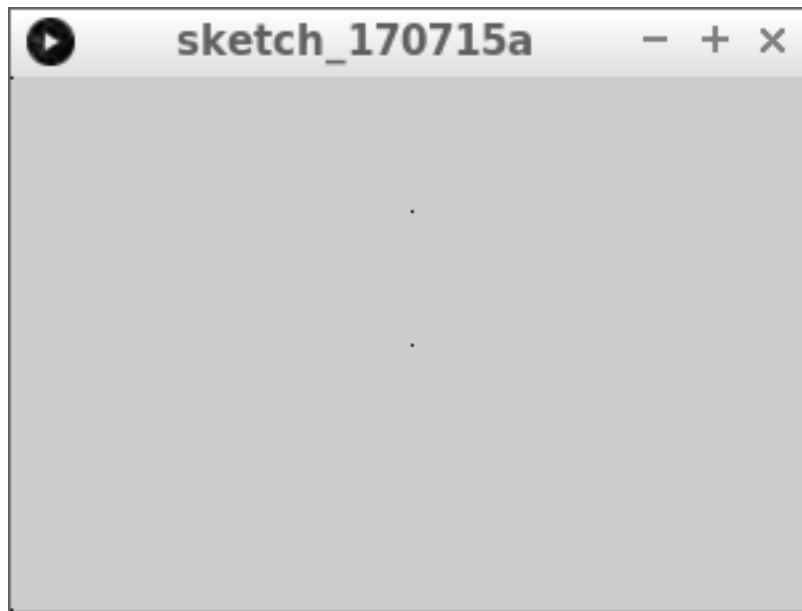


Figure 33: Opdracht 6

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(0, height - 1);
  point(width - 1, height - 1);
}
```

Opdracht 7

Run deze code:

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

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

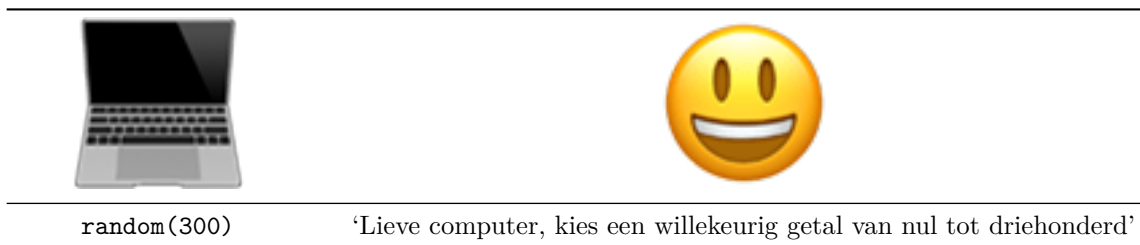
Wat zie je?

Oplossing 7



Figure 34: Oplossing 7

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



Opdracht 8



Figure 35: Opdracht 8

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.

Oplossing 8

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

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

Eindopdracht



Figure 36: Eindopdracht

Laat de computer willekeurig puntjes tekenen in het hele venster.