Figure 1: Boek 4

Contents

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

Het is nog een beetje een slordig boek. Er zitten tiepvauten in en de opmaak is ni`et altijd` *even moo*i.

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.

`fullScreen`

`fullScreen` is een functie waarmee je het venster van je programma net zo groot maakt als het beeldscherm van je computer.

Opdracht 1

Run deze code. Wat zie je?

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(100, 200, width / 4, height / 4);
}
```

	
`rect(100, 200, 300, 400)`	'Lieve computer, teken een rechthoek met `(100, 200)` als linkerbovenhoek, 300 pixels breed en 400 pixels hoog is.'
`width / 4`	'Lieve computer, vul hier het aantal pixels dat het scherm breed is, gedeeld door vier'
`height / 4`	'Lieve computer, vul hier het aantal pixels dat het scherm hoog is, gedeeld door vier'

Oplossing 1

Opdracht 2

Maak een rechthoek met de linkerbovenhoek in het midden, met een breedte van 200 en een hoogte van 100 pixels.

Oplossing 2

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 2, height / 2, 200, 100);
}
```

Opdracht 3

Zet nu de rechthoek in het midden van het scherm.

Figure 4: Oplossing 1

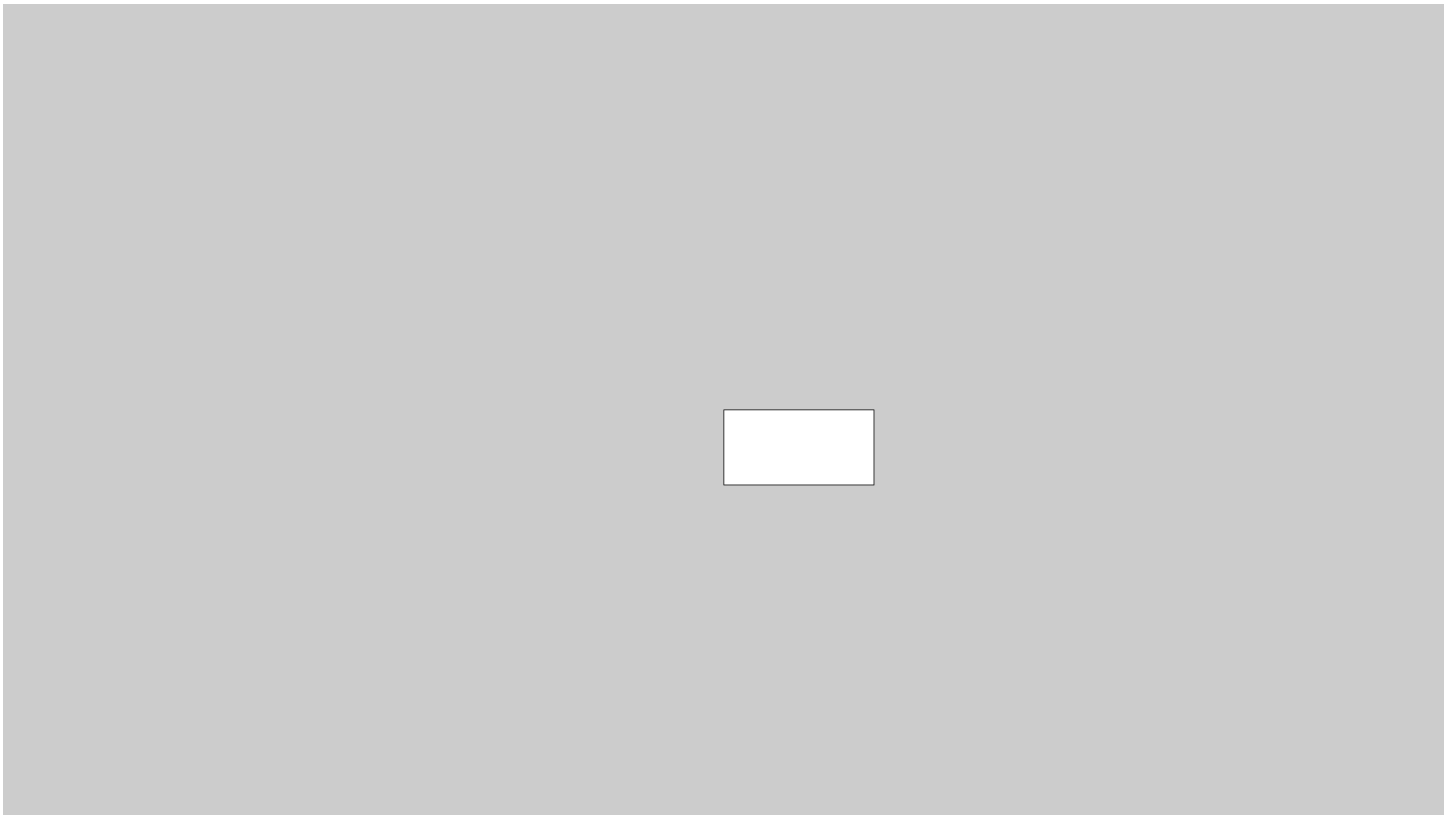


Figure 5: Opdracht 2

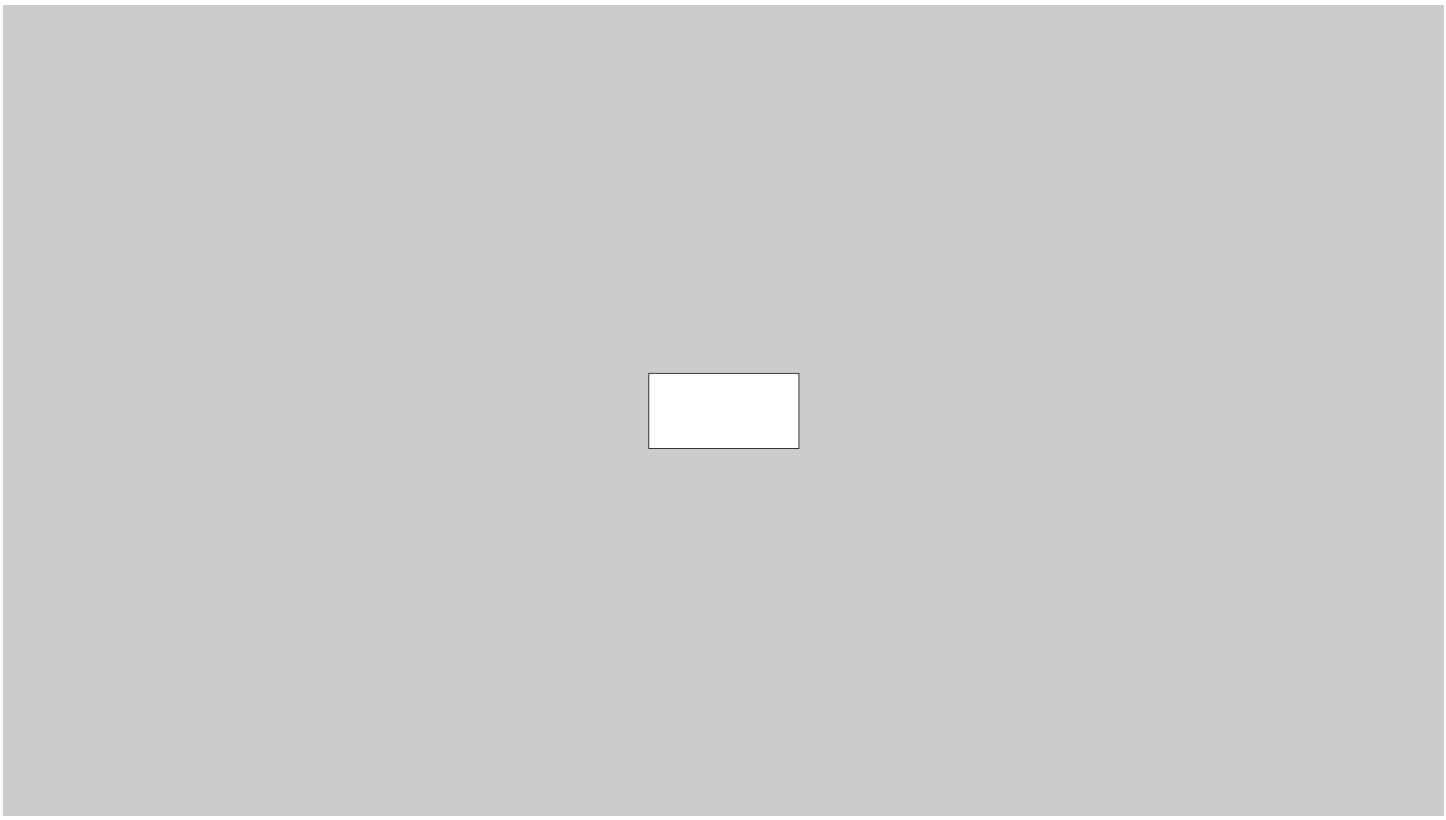


Figure 6: Opdracht 3

De rechthoek moet 100 naar links en 50 omhoog

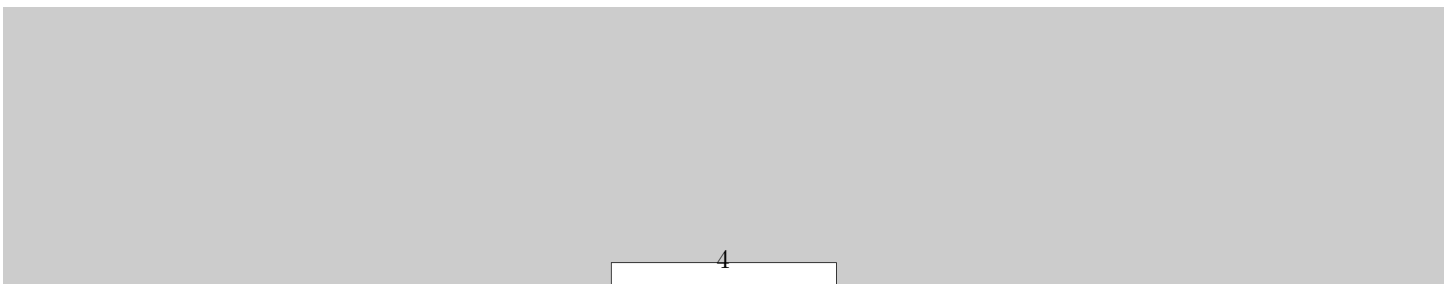
Oplossing 3

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 2 - 100, height / 2 - 50, 200, 100);
}
```

Opdracht 4

Zet een rechthoek in het midden van het scherm, met een breedte van 300 pixels en een hoogte van 400 pixels.



```
  fullScreen();
}

void draw()
{
  rect(width / 2 - 150, height / 2 - 200, 300, 400);
}
```

Opdracht 5

Zet een rechthoek in het midden van het scherm, die half zo breed is als het scherm, en 500 pixels hoog is.

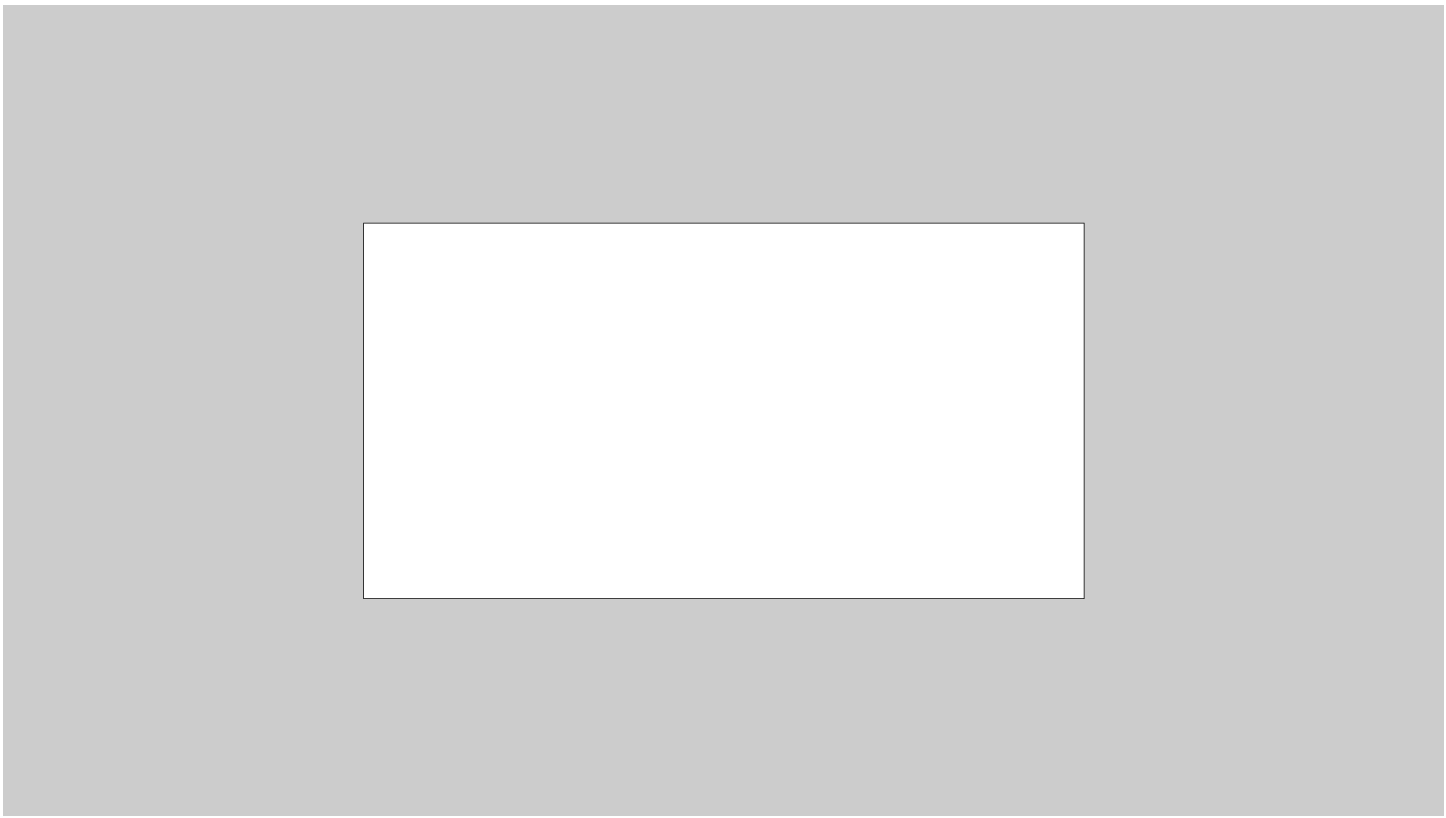


Figure 8: Opdracht 5

Oplossing 5

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 4, height / 2 - 250, width / 2, 500);
}
```

Eindopdracht

Zet een rechthoek in het midden van het scherm, die half zo breed en hoog is als het scherm.

PImage

In deze les gaan we met plaatjes werken!

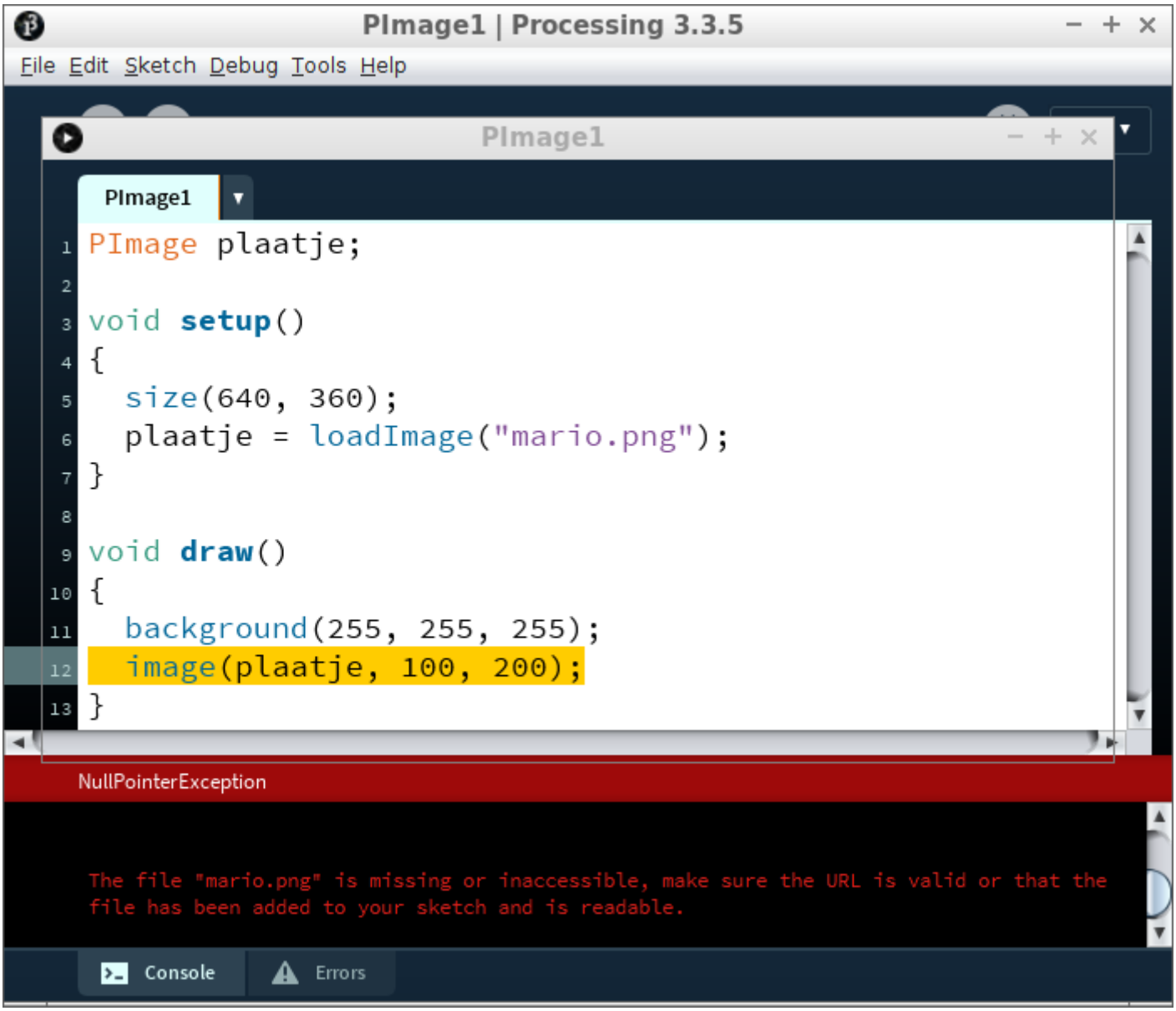


Figure 9: PImage1.png

Opdracht 1

Save deze code. Run deze code. Wat zie je?

```
PImage plaatje;

void setup()
{
  size(640, 360);
  plaatje = loadImage("mario.png");
}

void draw()
{
```

```
  background(255, 255, 255);
  image(plaatje, 100, 200);
}
```

Oplossing 1

Je krijgt een error!

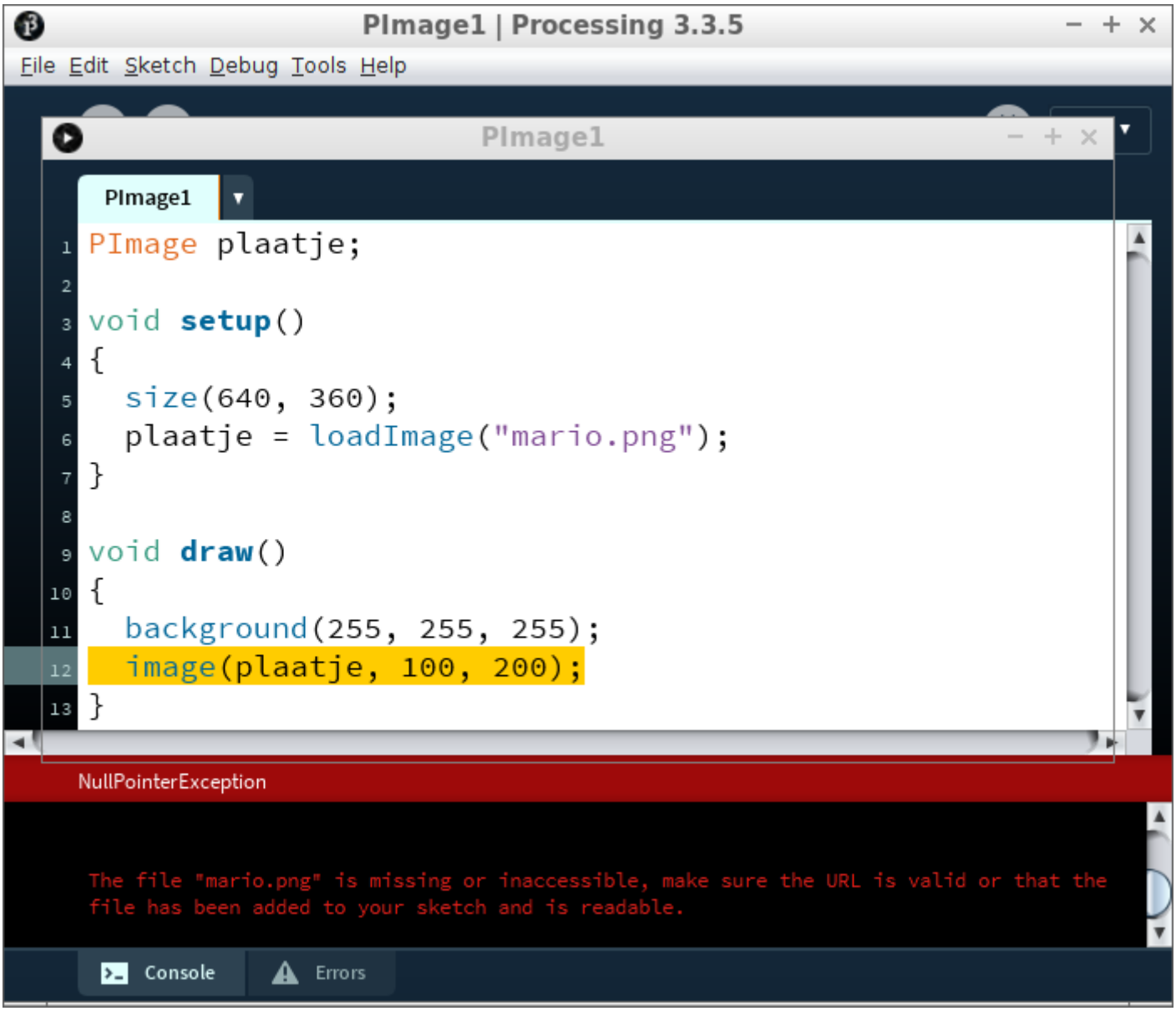


Figure 10: Oplossing 1

De computer zegt dat hij het plaatje niet kan vinden!

Opdracht 2

Ga naar https://github.com/richelbilderbeek/Dojo/blob/master/LessenProcessing/PImage/mario.png en download dit plaatje van Mario.

Figure 11: mario.png

Stop dit plaatje in een subfolder van waar je code staat.

Hier zie je een plaatje waarop staat waar de bestanden moeten staan:

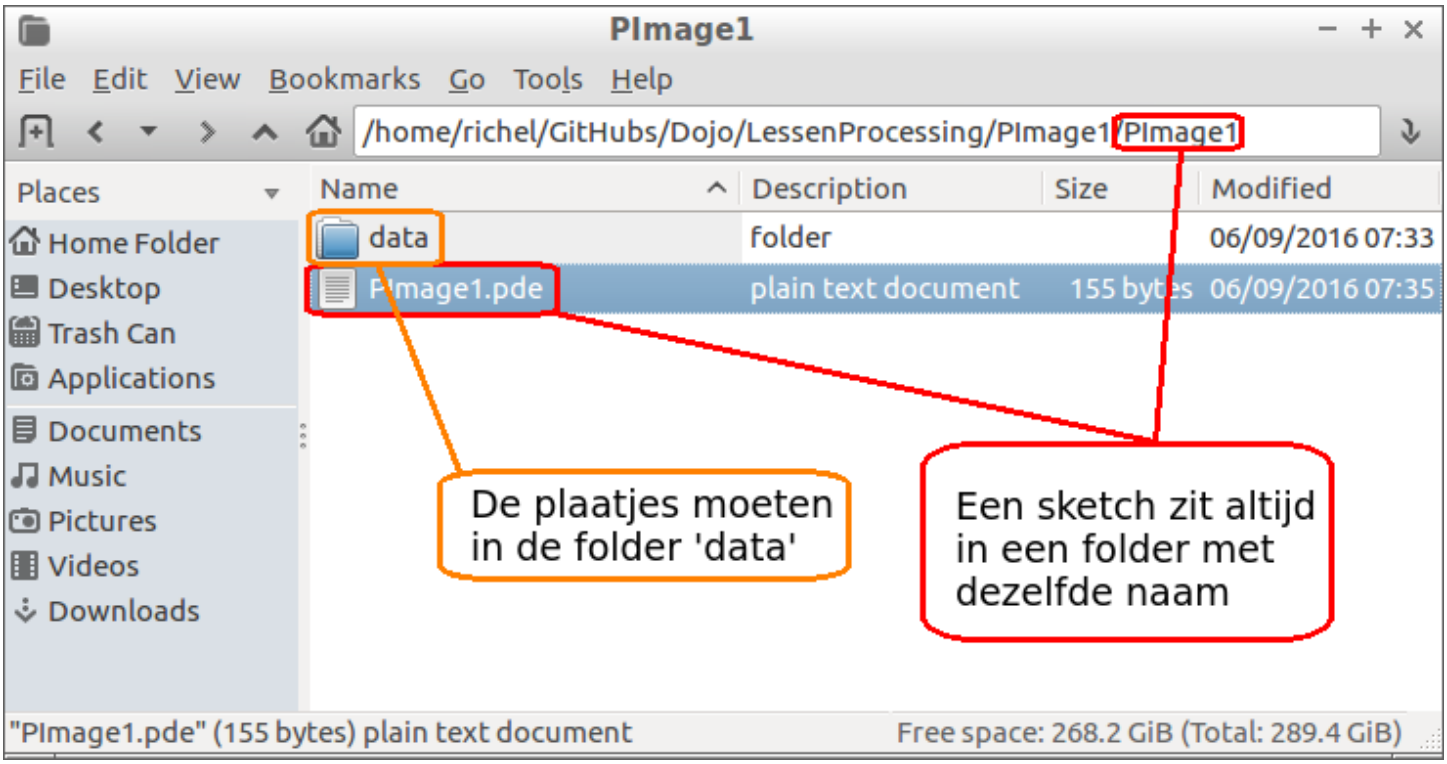


Figure 12: Folder structuur

- De sketch heet `PImage1.pde`. Daarom staat deze in de map `PImage1`. Deze kan je in Processing vinden onder `Schets -> Toon Schets Map`
- De sketch heeft een folder `data`. Hierin staat het plaatje, `mario.png`

Eindopdracht

Maak het programma full-screen, maak de achtergrond groen en zet het plaatje in het midden.

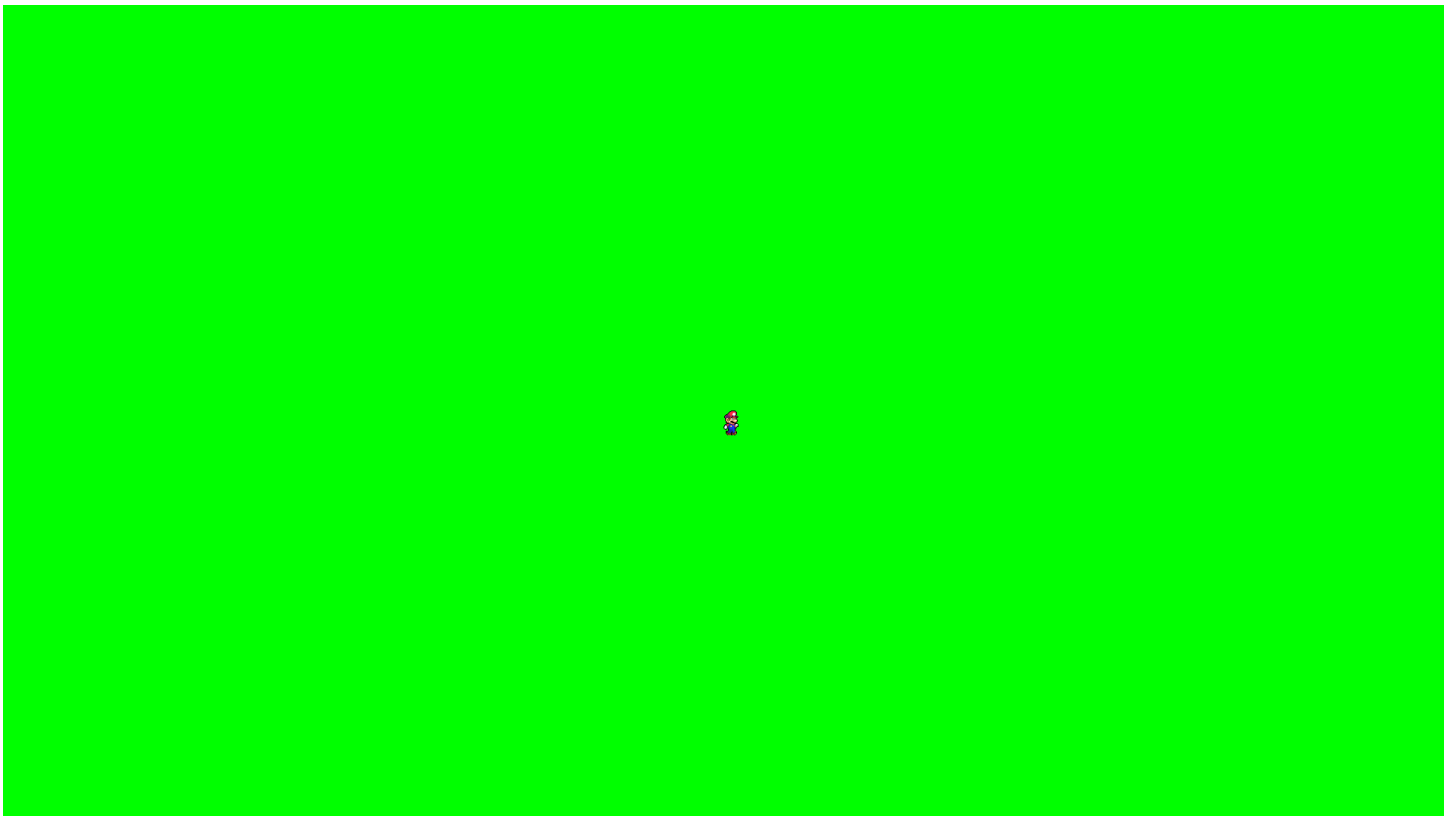


Figure 13: PImage eindopdracht

Zwaartekracht

In deze les gaan we zwaartekracht programmeren.

Het ziet er zo uit:

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

Opdracht 1

Wat doet deze code?

```
float x = 150;
float y = 100;
float snelheid_naar_rechts = 1;
float snelheid_omlaag = 1;

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

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + snelheid_naar_rechts;
  y = y + snelheid_omlaag;
  if (x > 275)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
```

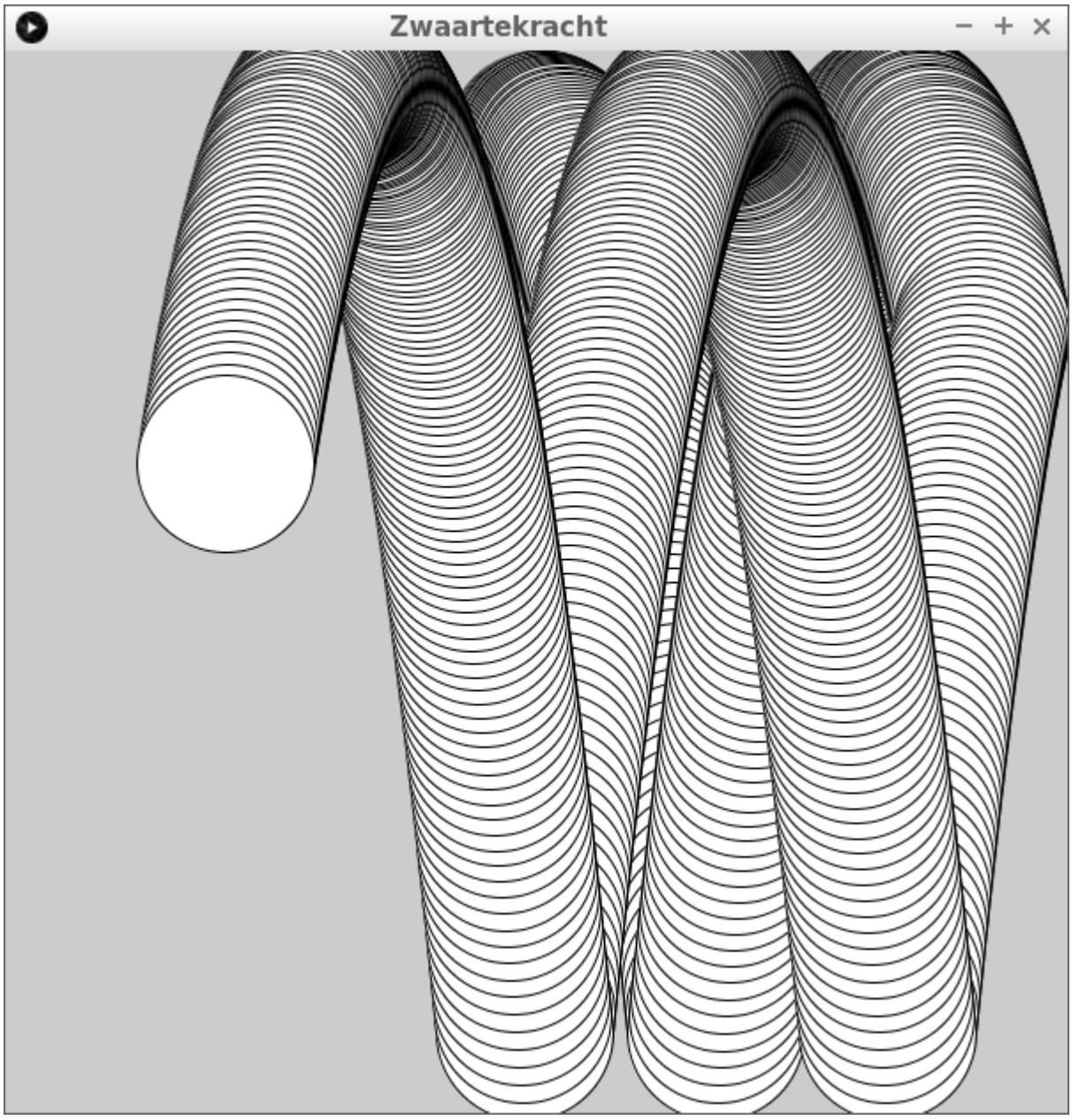


Figure 14: Zwaartekracht

```
  if (x < 25)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
  if (y > 175)
  {
    snelheid_omlaag = -snelheid_omlaag;
  }
}
```

Oplossing 1

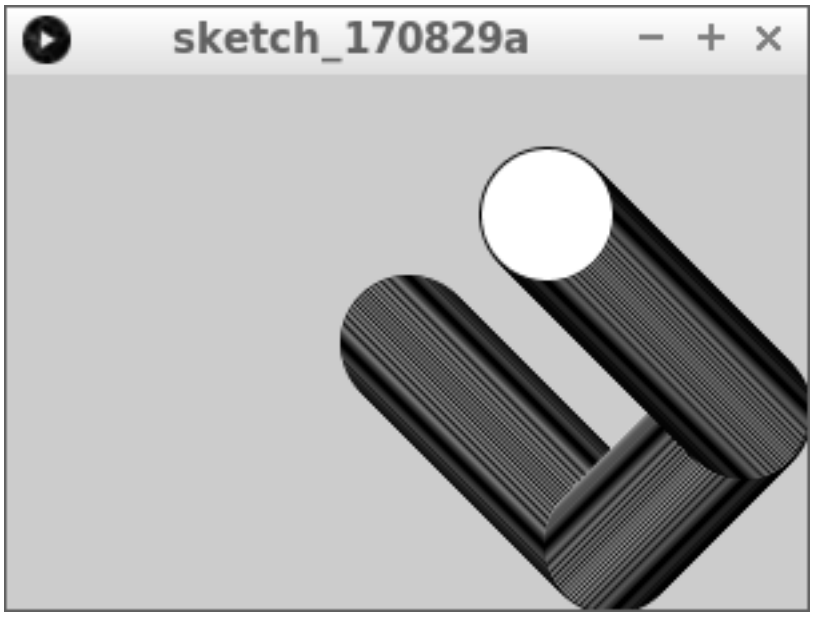


Figure 15: Zwaartekracht oplossing 1

Opdracht 2

Zorg dat het programma full-screen wordt

Oplossing 2

```
float x = 150;
float y = 100;
float snelheid_naar_rechts = 1;
float snelheid_omlaag = 1;

void setup()
{
  fullScreen();
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + snelheid_naar_rechts;
  y = y + snelheid_omlaag;
```

Figure 16: Zwaartekracht opdracht 2

```
  if (x > 275)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
  if (x < 25)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
  if (y > 175)
  {
    snelheid_omlaag = -snelheid_omlaag;
  }
}
```

Opdracht 3

Zorg dat de bal goed aan de onderkant stuitert.

Tip: vervang `175` door `height - 25`

Oplossing 3

```
float x = 150;
float y = 100;
float snelheid_naar_rechts = 1;
float snelheid_omlaag = 1;
```

Figure 17: Zwaartekracht opdracht 3

Opdracht 4

Zorg dat de bal goed aan de rechterkant stuitert.

Oplossing 4

```
float x = 150;
float y = 100;
float snelheid_naar_rechts = 1;
float snelheid_omlaag = 1;

void setup()
{
  fullScreen();
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + snelheid_naar_rechts;
  y = y + snelheid_omlaag;
  if (x > width - 25)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
  if (x < 25)
  {
    snelheid_naar_rechts = -snelheid_naar_rechts;
  }
```

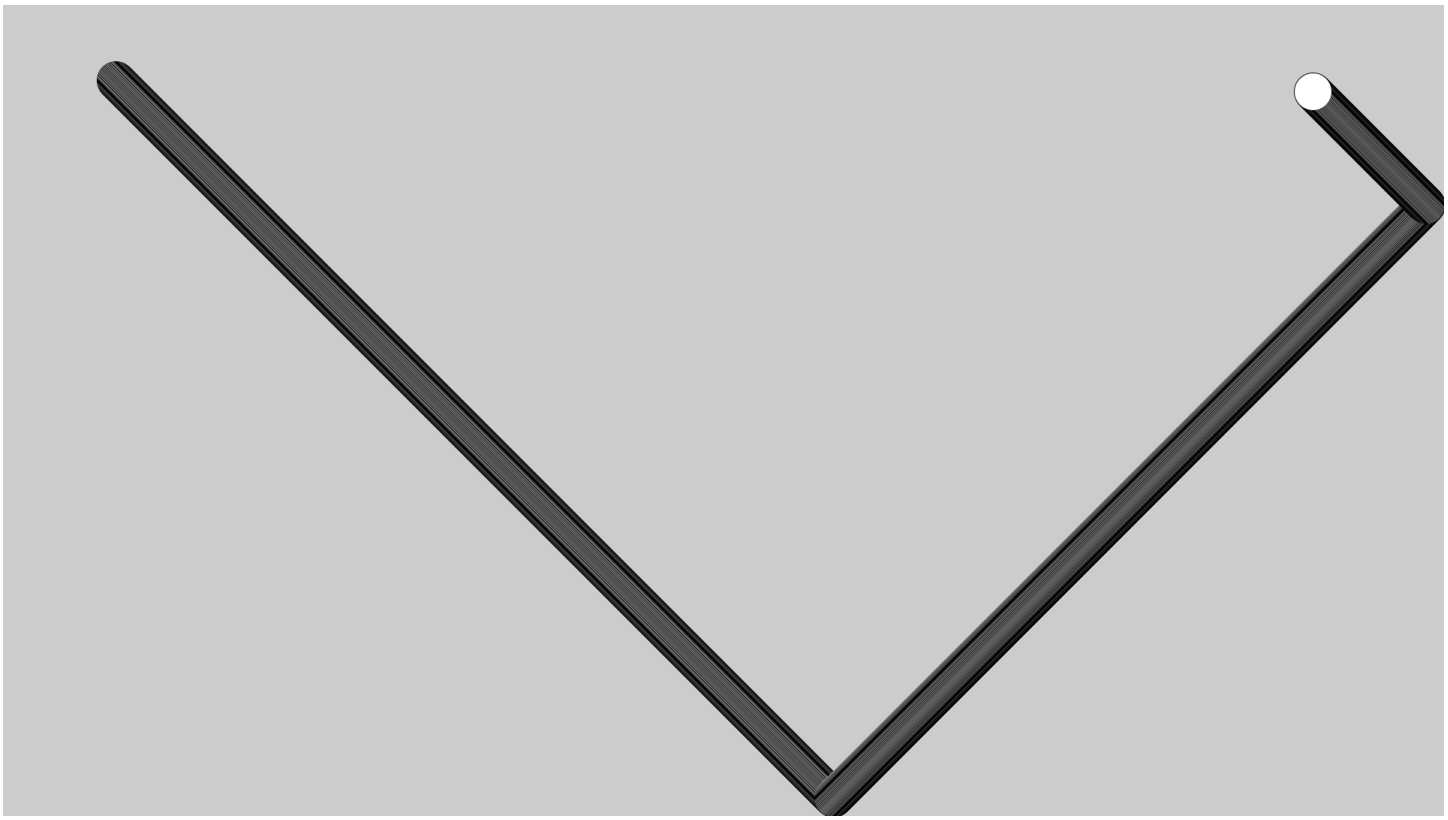


Figure 18: Zwaartekracht opdracht 4

```
  if (y > height - 25)
  {
    snelheid_omlaag = -snelheid_omlaag;
  }
}
```

Eindopdracht

Voeg onderaan de `draw` functie toe:

```
snelheid_omlaag += 0.1;
```

Maak de bal twee keer zo groot.

Als je figuur `Zwaartekracht eindopdracht ook goed` krijgt, krijg je ook een sticker.

Soms gebeuren er onverwachte dingen in programmeren!

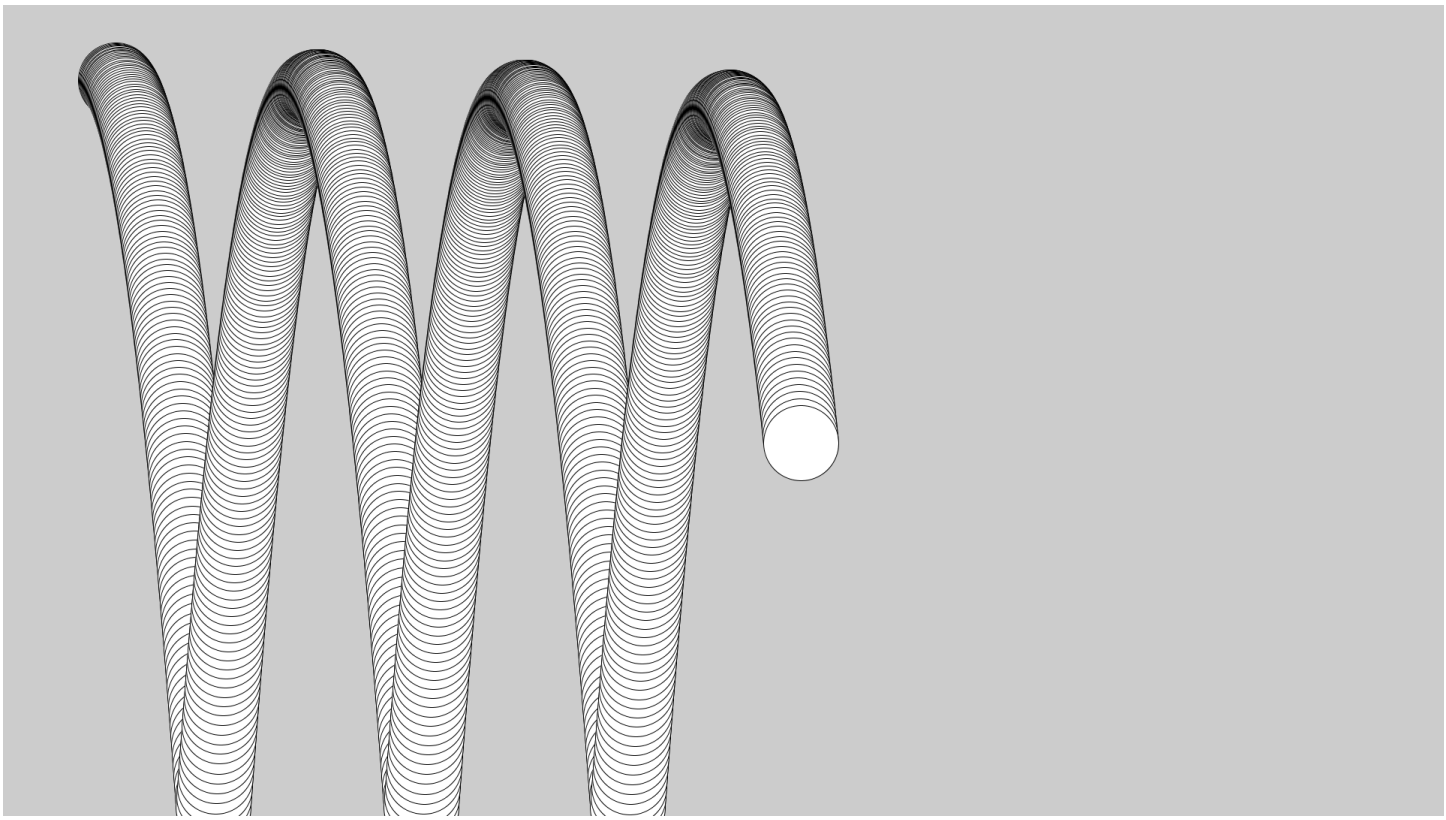


Figure 19: Zwaartekracht eindopdracht

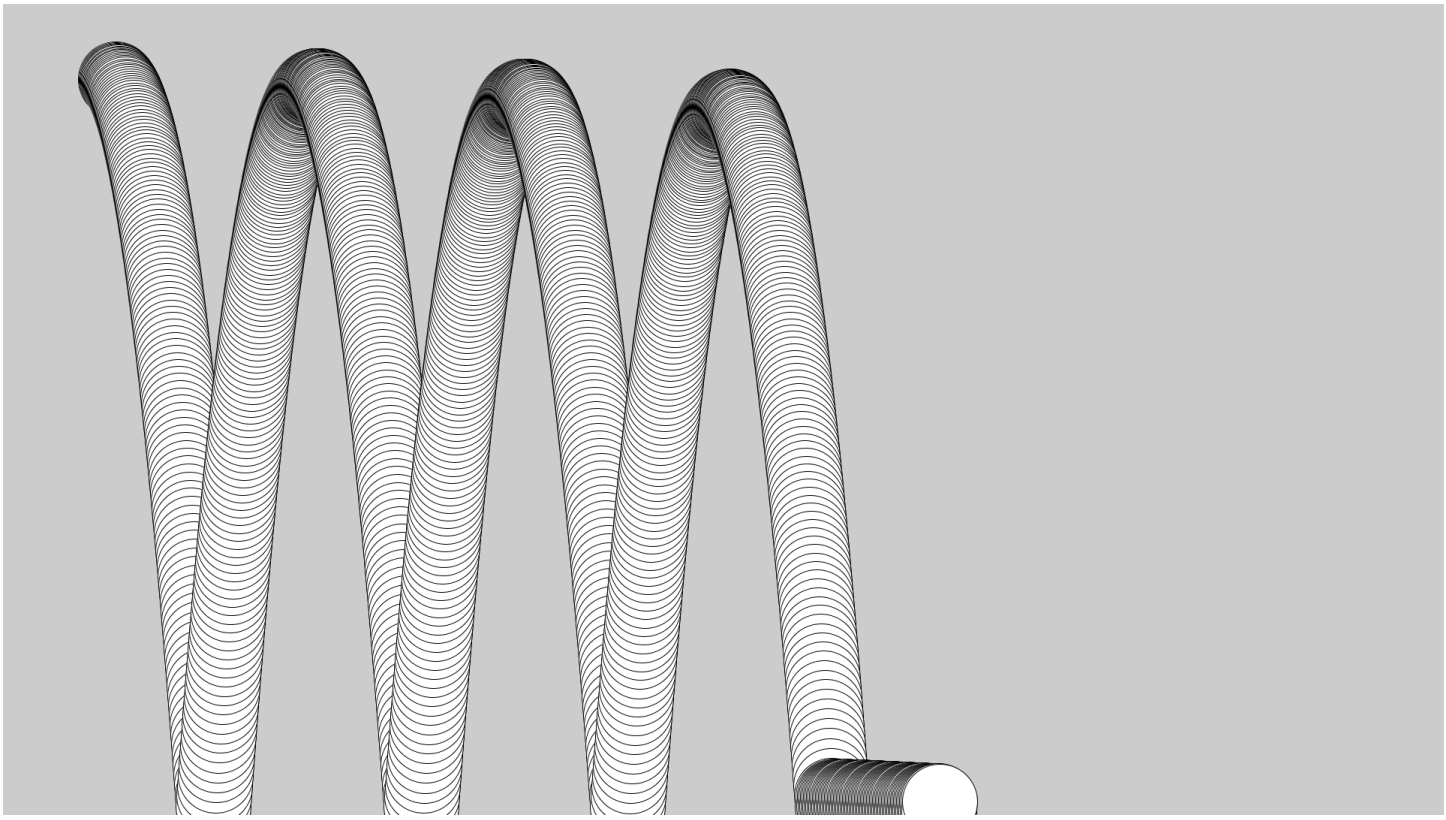


Figure 20: Zwaartekracht eindopdracht ook goed

Arrays1

Met arrays kun je de computer veel waardes laten onthouden: de coordinaten van kogels, meteorieten, vijanden.

Sla niet de eerste opdrachten over. Dit helpt je om arrays beter te snappen.

Opdracht 1

Run deze code. Wat doet het?

```
float x = 0;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x,25,50,50);
  x = x + 1;
  if (x > 625)
  {
    x = -25;
  }
}
```

Oplossing 1

Een bal die eeuwig naar rechts gaat!

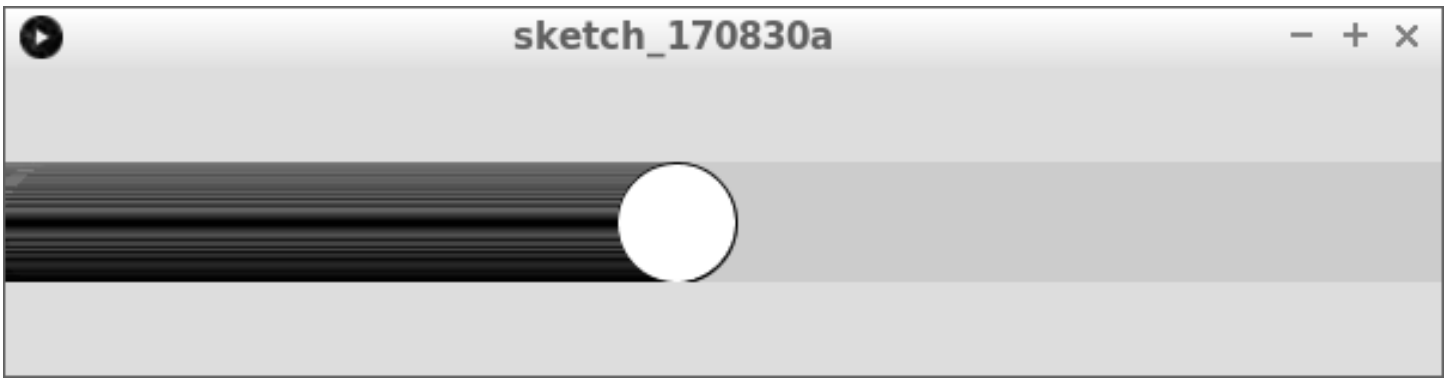


Figure 21: Oplossing 1

Opdracht 2

Zorg dat er een tweede bal bijkomt.

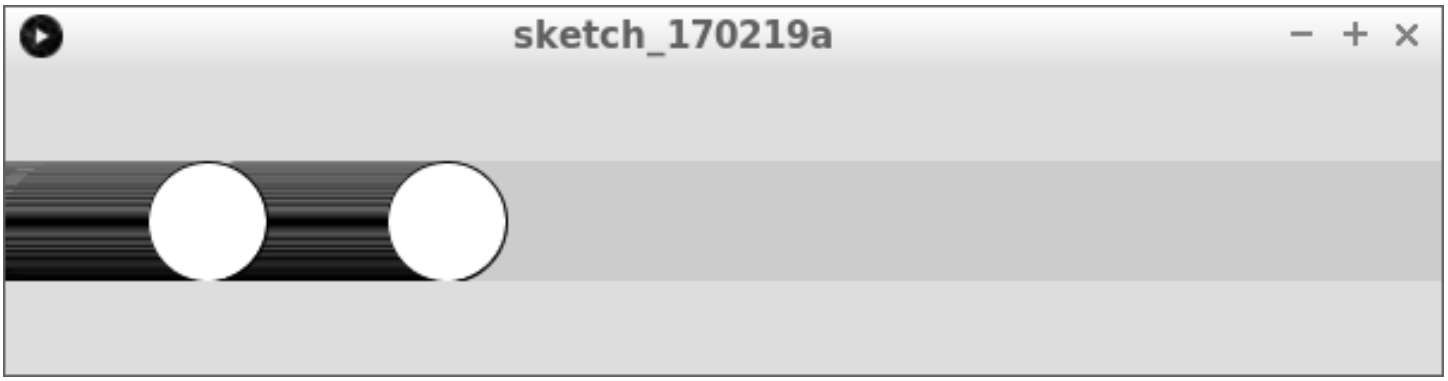


Figure 22: Twee ballen die eeuwig naar rechts gaan

Tip: verander de naam `x` naar `x1`

Maak dan een nieuwe variabele met de naam `x2`

Oplossing 2

```
float x1 = 0;
float x2 = 100;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x1,25,50,50);
  ellipse(x2,25,50,50);
  x1 = x1 + 1;
  x2 = x2 + 1;
  if (x1 > 625)
  {
    x1 = -25;
  }
  if (x2 > 625)
  {
    x2 = -25;
  }
}
```

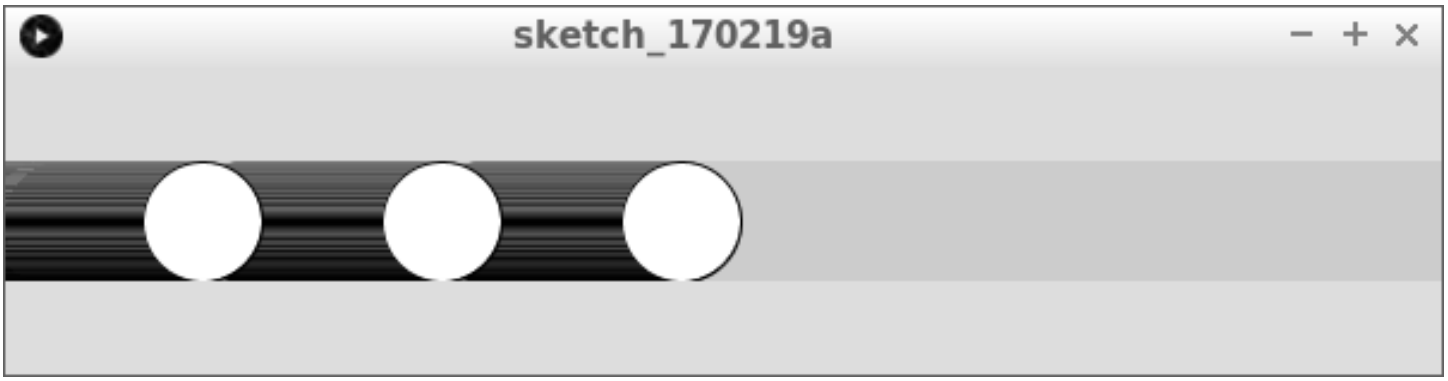


Figure 23: Drie ballen

Oplossing 3

```
float x1 = 0;
float x2 = 100;
float x3 = 200;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x1,25,50,50);
  ellipse(x2,25,50,50);
  ellipse(x3,25,50,50);
  x1 = x1 + 1;
  x2 = x2 + 1;
  x3 = x3 + 1;
  if (x1 > 625)
  {
    x1 = -25;
  }
  if (x2 > 625)
  {
    x2 = -25;
  }
  if (x3 > 625)
  {
    x3 = -25;
  }
}
```

Dit was weer zeven regels extra werk

Dit kan slimmer, met arrays!

Figure 24: Kast met laatje

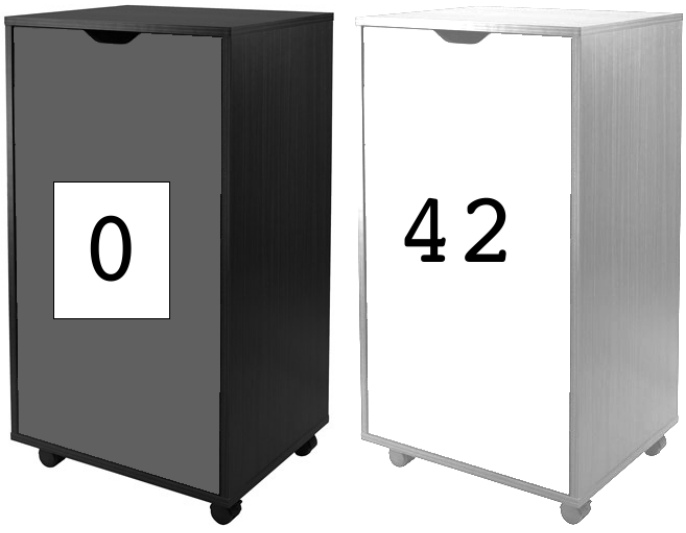


Figure 25: Kast met genummerde laatjes

 Het eerste laatje van een array heeft nummer nul

Het laatje heeft nummer *nul* (links) en in het laatje zit het getal tweeenveertig.

	
`plek in array met index nul`	‘het eerste plekje in de array’

Werken met een array met een laatje

Figure 26: Array met naam ‘geheime_getallen’ en een laatje

Stel we willen een array maken van gebroken getallen (`floats`) met de naam `geheime_getallen`, dan moeten we boven de `setup` het volgende typen:

```
float[] geheime_getallen;
```

Met deze regel maak je array met de naam `geheime_getallen`

	
`float[] geheime_getallen`	‘Lieve computer, onthoud keiveel gebroken getallen met de naam `geheime_getallen`’

Er is nog niet gezegd *hoeveel* gebroken getallen dat zijn. Vaak wordt de `setup` functie gebruikt om te zeggen hoeveel getallen er onthouden moeten worden:

```
geheime_getallen = new float[1];
```

Hiermee maak je de array `geheime_getallen` 1 laatje groot.

	
`geheime_getallen = new float[1]`	'Lieve computer, maak `geheime_getallen` 1 laatje groot''

Om de kast met de laatjes na te maken, kun je de volgende code gebruiken:

```
geheime_getallen[0] = 42;
```

Hierdoor stop je het getal 42 op het eerste plekje in de array.

	
`geheime_getallen[0] = 42`	'Lieve computer, stop het getal 42 in het eerste plekje van de array `geheime_getallen`'

Je zou ook de waarde in de laatjes kunnen lezen:

```
float x = geheime_getallen[0];
```

Hiermee lees je het eerste plekje (het laatje met index nul) en stop je dat in `x`.

	
`float x = geheime_getallen[0]`	'Lieve computer, kijk wat er in laatje met index nul zit en onthoud dat als `x`'

Alles bij elkaar krijg je dit programma:

```
float[] geheime_getallen;

void setup()
{
  size(400,400);
  geheime_getallen = new float[1];
  geheime_getallen[0] = 42;
}

void draw()
{
  float x = geheime_getallen[0];
```

```
  ellipse(x,200,300,400);
}
```

Dit programma ziet er niet erg mooi uit. Het is bedoeld om je te laten hoe je arrays maakt, vult en leest.

Opdracht 4

Run onderstaande code

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[3];
  xs[0] = 0;
  xs[1] = 100;
  xs[2] = 200;
}

void draw()
{
  ellipse(xs[0],25,50,50);
  ellipse(xs[1],25,50,50);
  ellipse(xs[2],25,50,50);
  xs[0] = xs[0] + 1;
  xs[1] = xs[1] + 1;
  xs[2] = xs[2] + 1;
  if (xs[0] > 625)
  {
    xs[0] = -25;
  }
  if (xs[1] > 625)
  {
    xs[1] = -25;
  }
  if (xs[2] > 625)
  {
    xs[2] = -25;
  }
}
```

Oplossing 4

Hee, hetzelfde als net!

Opdracht 5

Run deze code:

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[3];
  for (int i=0; i<3; ++i)
  {
```

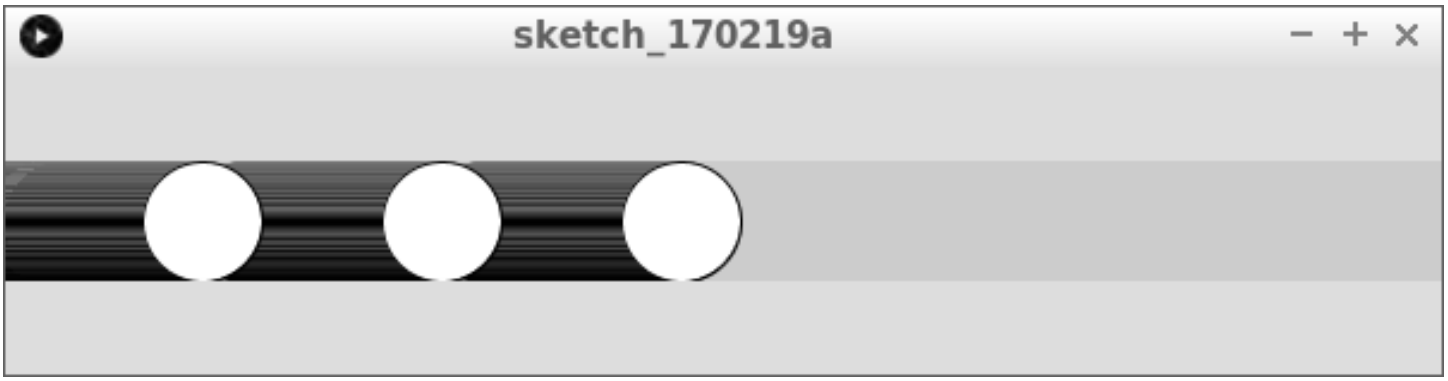


Figure 27: Oplossing 4

```
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<3; ++i)
  {
    ellipse(xs[i],25,50,50);
    xs[i] = xs[i] + 1;
    if (xs[i] > 625)
    {
      xs[i] = -25;
    }
  }
}
```

	
`for (int i=0; i<3; ++i) {}`	'Lieve computer, doe wat tussen accolades staat met waarden van `i` van `0` tot `3` in stapjes van `1`'

 Ik ben een domme computer

```
xs[0] =   0;
xs[1] = 100;
xs[2] = 200;
```

 Ik ben een slimme computer

```
for (int i=0; i<3; ++i)
{
```

Oplossing 5

Hee, hetzelfde als net!

Opdracht 6

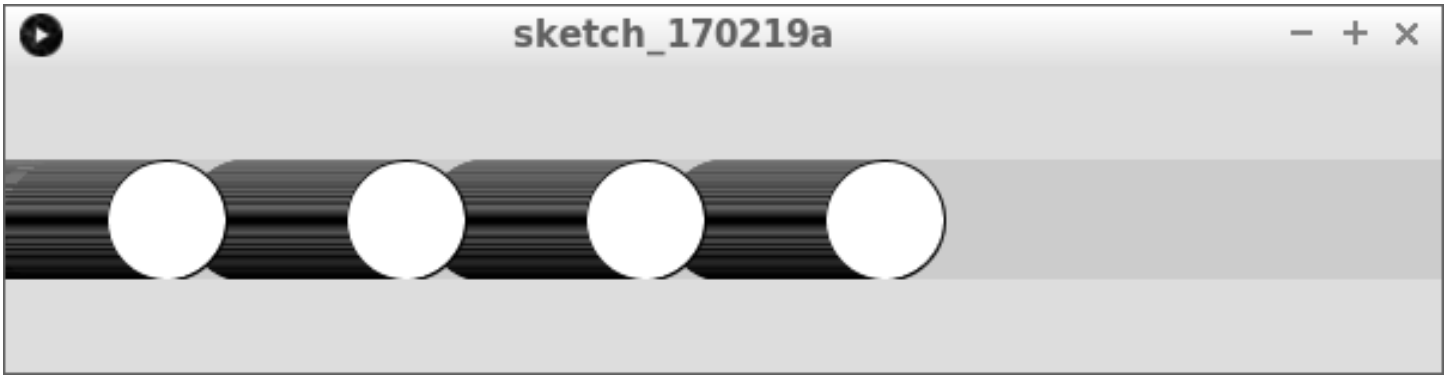


Figure 28: Vier ballen

Maak nu een vierde bal erbij.

Tip: maak van een `3` een `4`

Oplossing 6

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    ellipse(xs[i], 25, 50, 50);
    xs[i] = xs[i] + 1;
    if (xs[i] > 625)
    {
      xs[i] = -25;
    }
  }
}
```

Opdracht 7

Maak nu het programma full-screen. Laat de ballen als ze rechts het scherm uitgaan, weer links beginnen. Gebruik hiervoor `width`

Figure 29: Opdracht 7

Oplossing 7

```
float[] xs;

void setup()
{
  fullScreen();
  xs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    ellipse(xs[i], 25, 50, 50);
    xs[i] = xs[i] + 1;
    if (xs[i] > width + 25)
    {
      xs[i] = -25;
    }
```

```
  }
}
```

Eindopdracht

Figure 30: Eindopdracht

Maak de code nu zo dat:

- Er zes ballen zijn
- De ballen eeuwig naar links gaan

Arrays2

Met arrays kun je de computer veel waardes laten onthouden: de coordinaten van kogels, meteorieten, vijanden.

In deze les gaan we leren

- waarom je arrays nodig hebt
- wat arrays zijn
- hoe je een array met een element gebruikt

Zo gaat het eruit zien:

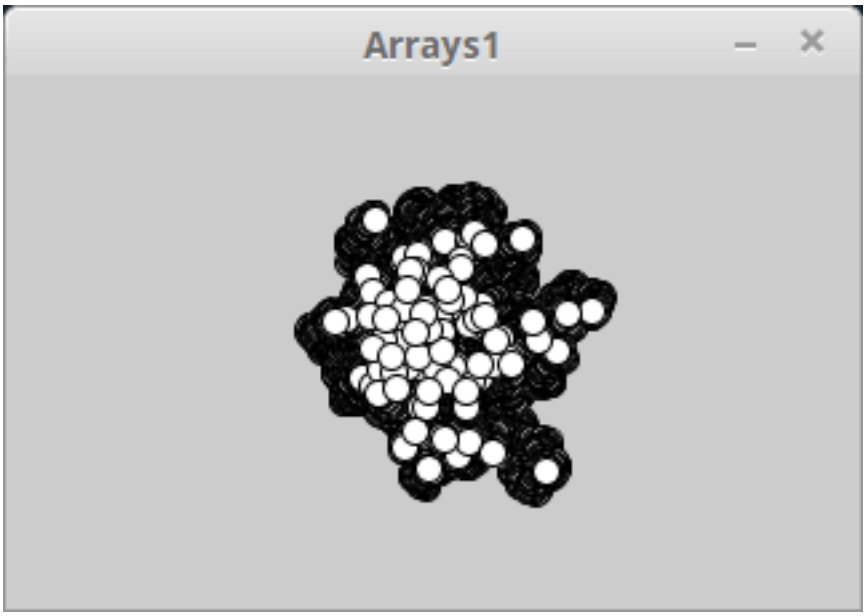


Figure 31: Honderd rookdeeltjes

Een rookdeeltje

Je bent bezig een simulatie te maken: je wilt allemaal rookdeeltjes laten bewegen op het scherm.

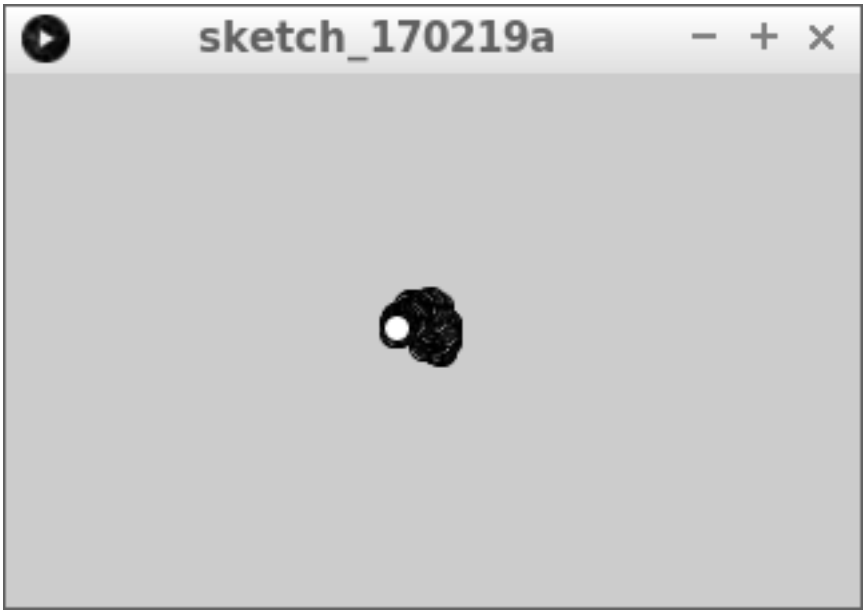


Figure 32: Een rookdeeltje

Dit is je code:

```
float x = 160;
float y = 100;

void setup()
{
  size(320, 200);
}

void draw()
{
  x += random(-1,1);
  y += random(-1,1);
  ellipse(x, y, 10, 10);
}
```

Dit is wat de code betekent:

- `float x = 160`: 'Lieve computer, onthoudt een gebroken getal met de naam `x`, met als beginwaarde 160'. Dit wordt de x coordinaat van het eerste rookdeeltje
- `float y = 100`: 'Lieve computer, onthoudt een gebroken getal met de naam `y`, met als beginwaarde 100'. Dit wordt de y coordinaat van het eerste rookdeeltje
- `void setup() {}`: de klaarzet functie. Bij het opstarten wordt de code tussen de accolates een keer uitgevoerd
- `size(320, 200)`: maak een venster van 320 pixels breed en 200 pixels hoog
- `void draw() {}`: de teken functie. De code tussen de accolates wordt oneindig vaak uitgevoerd
- `x += random(-1,1)`: verander de waarde van `x` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig horizontaal bewegen
- `y += random(-1,1)`: verander de waarde van `y` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig verticaal bewegen
- `ellipse(x, y, 10, 10)`: teken een ovaal met als middelpunt (`x`, `y`) met breedte 10 en hoogte 10. Teken het eerste rookdeeltje

Vragen

- 1. Zorg dat er een tweede rookdeeltje bijkomt
- 2. Hoeveel regels code kost het ongeveer om honderd rookdeeltjes te programmeren?

Oplossing

- 1. Dit ziet er zo uit:

```
float x1 = 160;
float y1 = 100;
float x2 = 160;
float y2 = 100;

void setup()
{
  size(320, 200);
}

void draw()
{
  x1 += random(-1,1);
  y1 += random(-1,1);
  ellipse(x1, y1, 10, 10);
  x2 += random(-1,1);
  y2 += random(-1,1);
  ellipse(x2, y2, 10, 10);
```

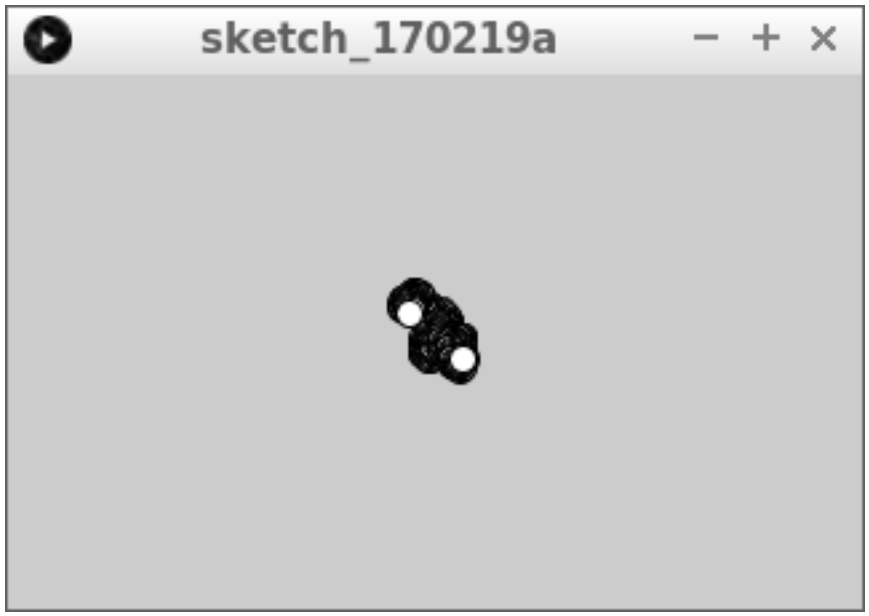


Figure 33: Twee rookdeeltjes

```
}
```

- 2. Dit kostte vijf regels

Twee rookdeeltjes met een array, zonder for loop

Om naar honderd rookdeeltjes te komen, gaan we eerst de code omschrijven.

Opdracht

Verander de code van 'Twee rookdeeltjes':

- gebruik een array `xs`, in plaats van `x1` en `x2`
- array `xs` is twee laatjes groot
- gebruik een array `ys`, in plaats van `y1` en `y2`
- array `ys` is twee laatjes groot
- In arrays `xs` zit een `160` in het eerste en tweede laatje
- In arrays `ys` zit een `100` in het eerste en tweede laatje
- gebruik nog geen for loop

Oplossing

```
float[] xs;
float[] ys;

void setup()
{
  size(320, 200);
  xs = new float[2];
  ys = new float[2];
  xs[0] = 160;
  xs[1] = 160;
  ys[0] = 100;
  ys[1] = 100;
}
```

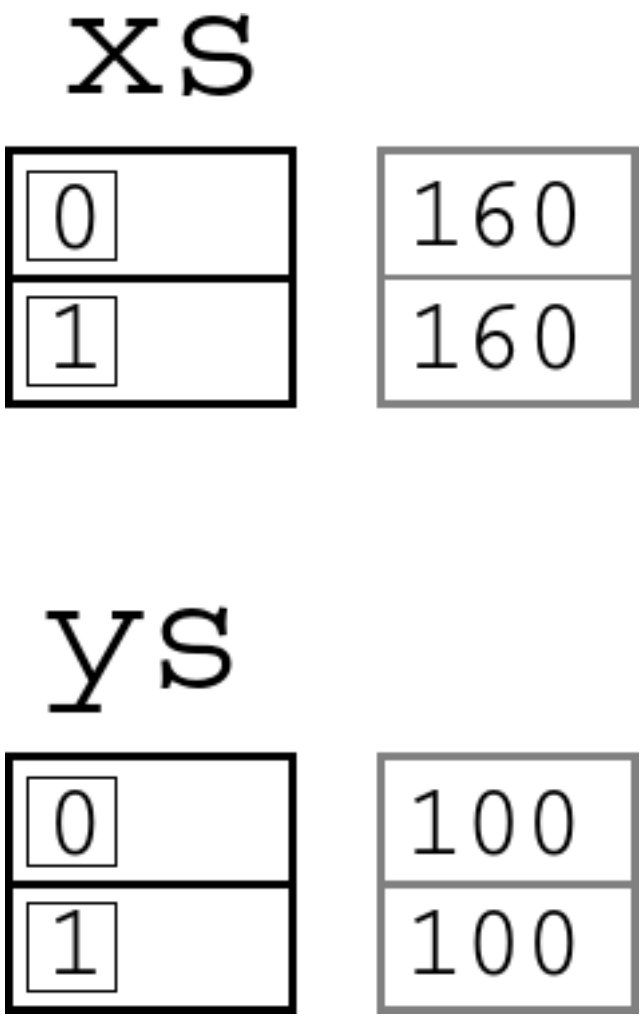


Figure 34: Array van ‘Twee rookdeeltjes met een array, zonder for loop’

```
void draw()
{
  xs[0] += random(-1,1);
  ys[0] += random(-1,1);
  ellipse(xs[0], ys[0], 10, 10);
  xs[1] += random(-1,1);
  ys[1] += random(-1,1);
  ellipse(xs[1], ys[1], 10, 10);
}
```

Drie rookdeeltjes

Nu laten we de code een for loop gebruiken.

Opdracht

Figure 35: Drie rookdeeltjes

Verander de code van 'Twee rookdeeltjes met een array, zonder for loop':

- Gebruik een for-loop
- Maak er drie rookdeeltjes van

Oplossing

```
float[] xs;
float[] ys;

void setup()
{
  size(320, 200);
  xs = new float[3];
  ys = new float[3];
  for (int i=0; i<3; ++i)
  {
    xs[i] = 160;
    ys[i] = 100;
```

```
  }
}

void draw()
{
  for (int i=0; i<3; ++i)
  {
    xs[i] += random(-1,1);
    ys[i] += random(-1,1);
    ellipse(xs[i], ys[i], 10, 10);
  }
}
```

Vier rookdeeltjes

We gaan weer een stapje verder: nu geven we de randen van de rookdeeltjes een rode kleur

Figure 36: Vier rookdeeltjes

Opdracht

- Gebruik nu vier rookdeeltjes
- Maak een derde array genaamd `rs`
- In `rs` zit de roodheid van de rookdeeltjes
- In `rs` moeten de getallen `0`, `64`, `128` en `196` komen. Tip: dit is de tafel van `64`. Om te vermenigvuldigen, gebruik een sterretje (`*`)
- De roodheid moet ook steeds een meer of minder worden
- De rand van het eerste rookdeeltje, moet de eerste roodheid krijgen. Tip: gebruik `stroke`

Oplossing

```
float[] xs;
float[] ys;
float[] rs; //Roodwaarden

void setup()
{
```

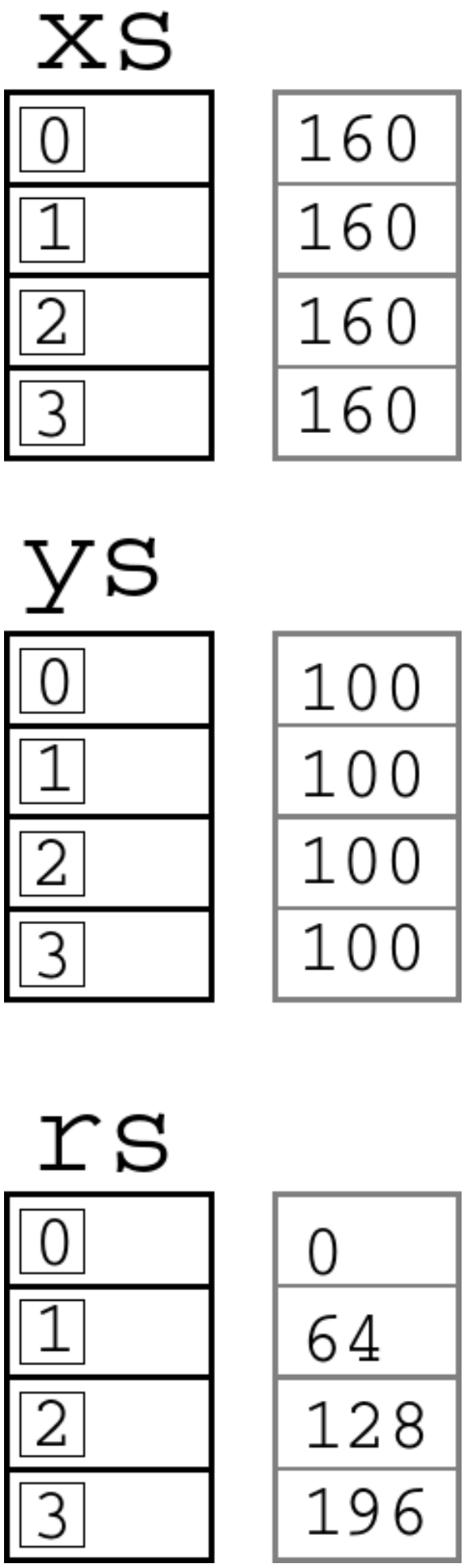


Figure 37: Arrays bij opdracht ‘Vier rookdeeltjes’

```
  size(320, 200);
  xs = new float[4];
  ys = new float[4];
  rs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = 160;
    ys[i] = 100;
    rs[i] = i * 64;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    xs[i] += random(-1,1);
    ys[i] += random(-1,1);
    rs[i] += random(-1,1);
    stroke(rs[i], 0, 0);
    ellipse(xs[i], ys[i], 10, 10);
  }
}
```

Eindopdracht

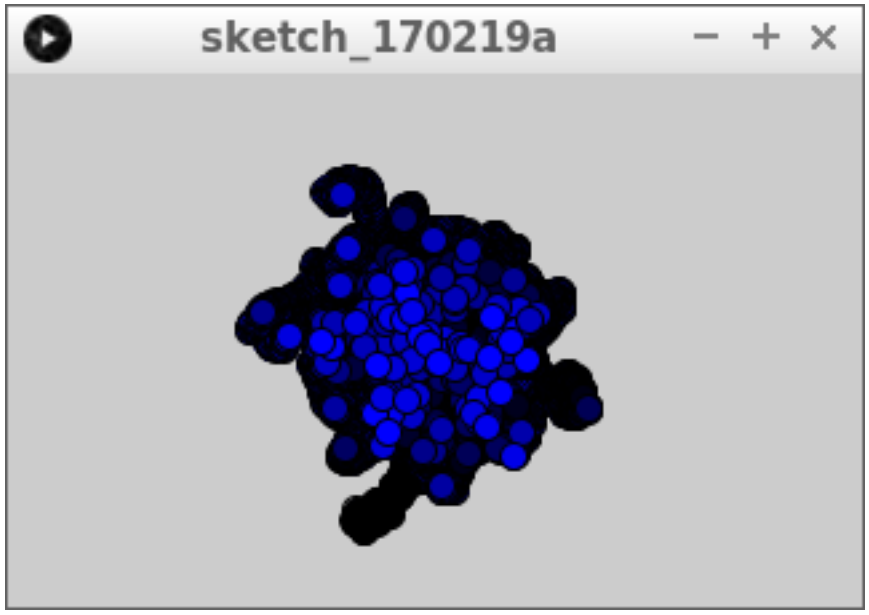


Figure 38: Arrays2 eindopdracht

Maak nu de code zo dat:

- er 256 rookdeeltjes komen.
- elk rookdeeltje heeft en eigen *blauw*waarde
- het eerste rookdeeltje heeft een blauwwaarde van nul. Het tweede rookdeeltje heeft een blauwwaarde van een. Het derde rookdeeltje heeft een blauwwaarde van twee. Enzovoorts
- noem de array waarin de blauwwaarden staan `bs`
- niet de rand, maar de vulkleur is blauw (tip: `fill`)