

Figure 1: Boek 4

Contents

Voorwoord	1
fullScreen	2
PImage	6
Zwaartekracht	9
Arrays1	16
Arrays2	27

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.

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

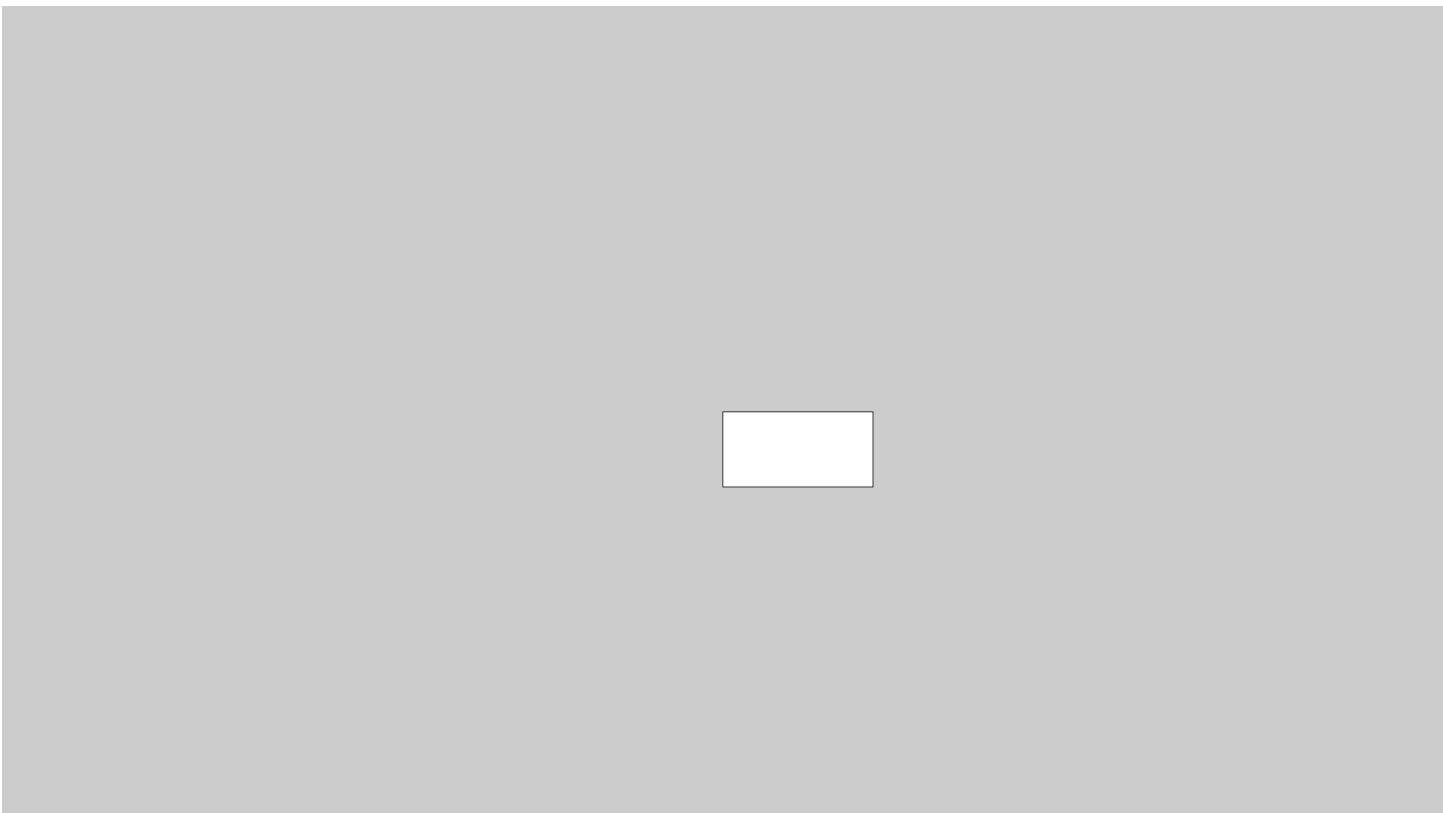


Figure 5: Opdracht 2

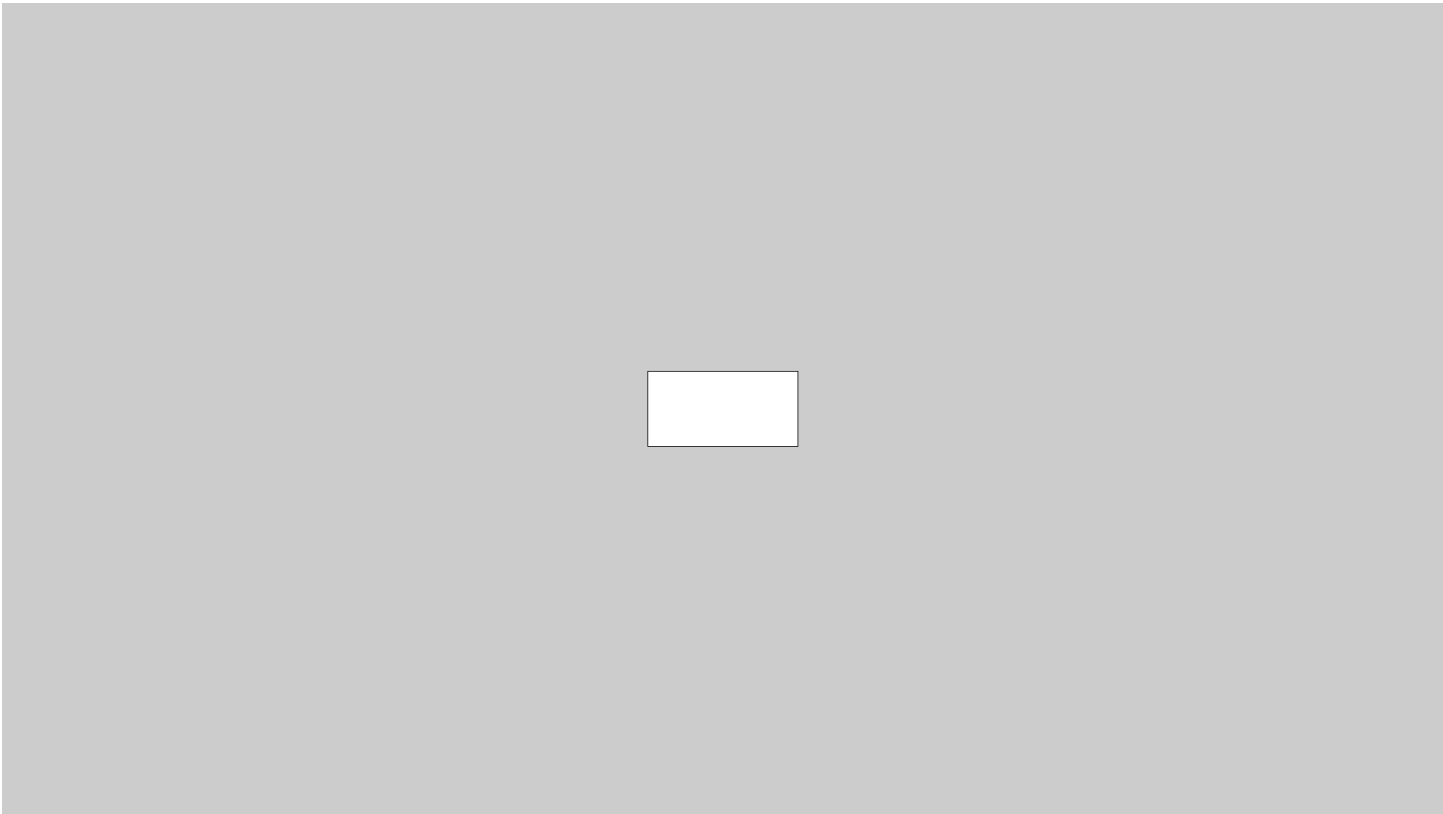


Figure 6: Opdracht 3



De rechthoek moet 100 naar links en 50 omhoog

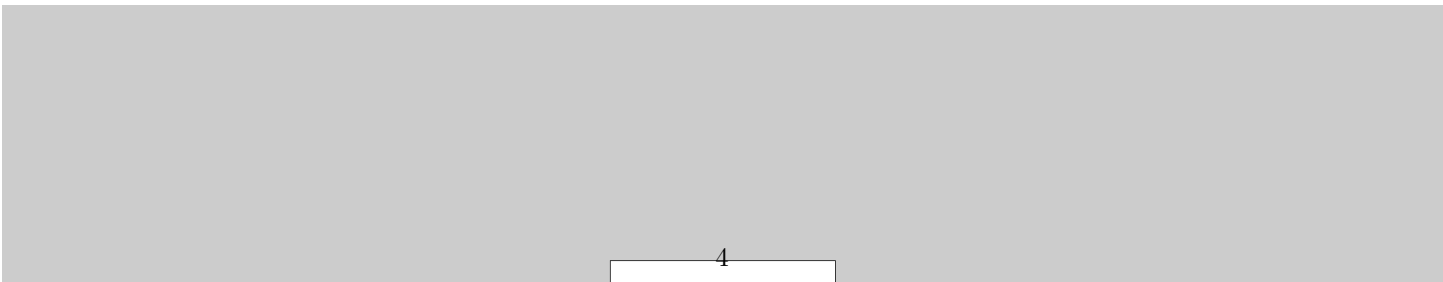
Oplossing 3

```
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.

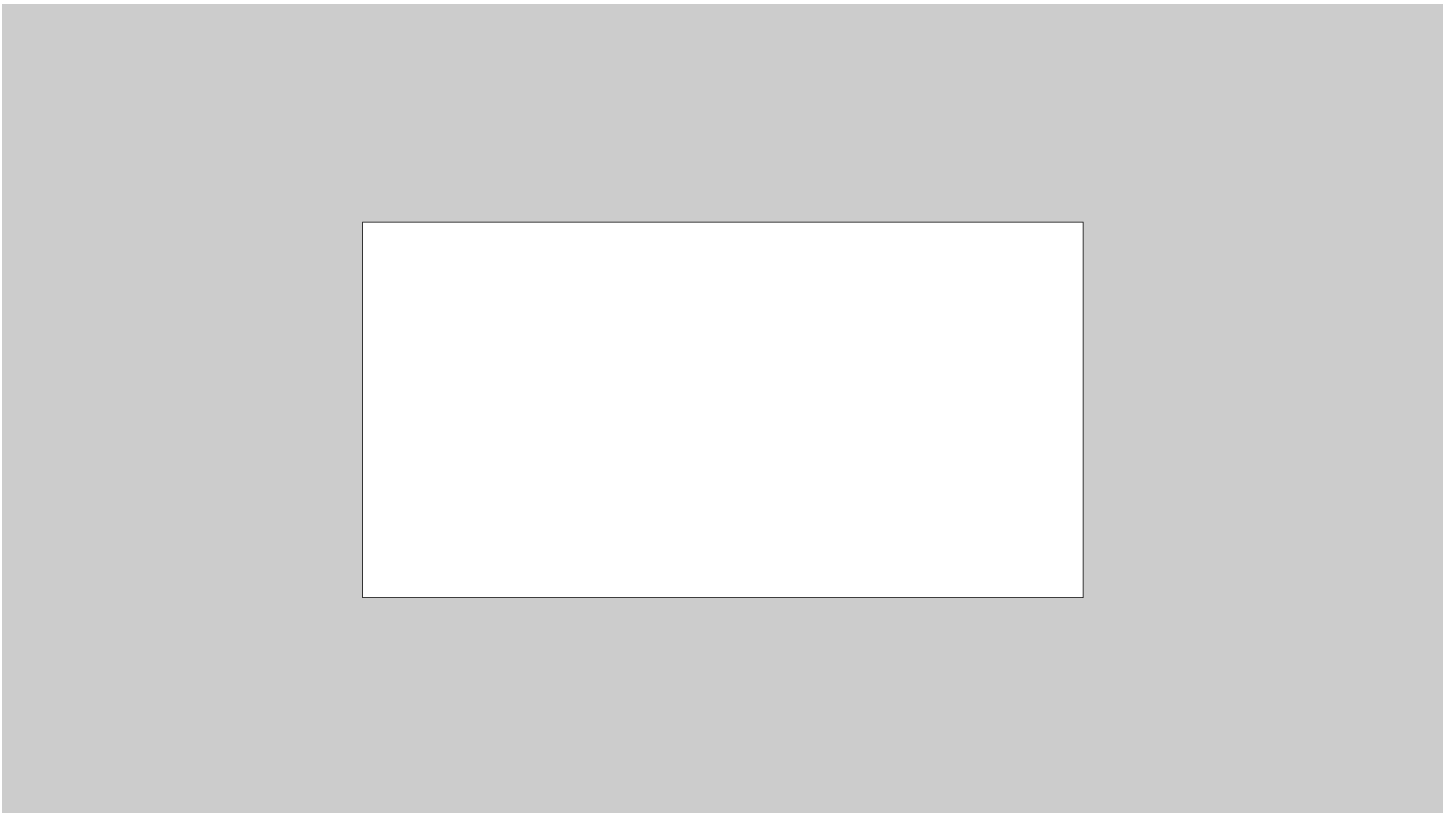


Figure 8: Opdracht 5

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!

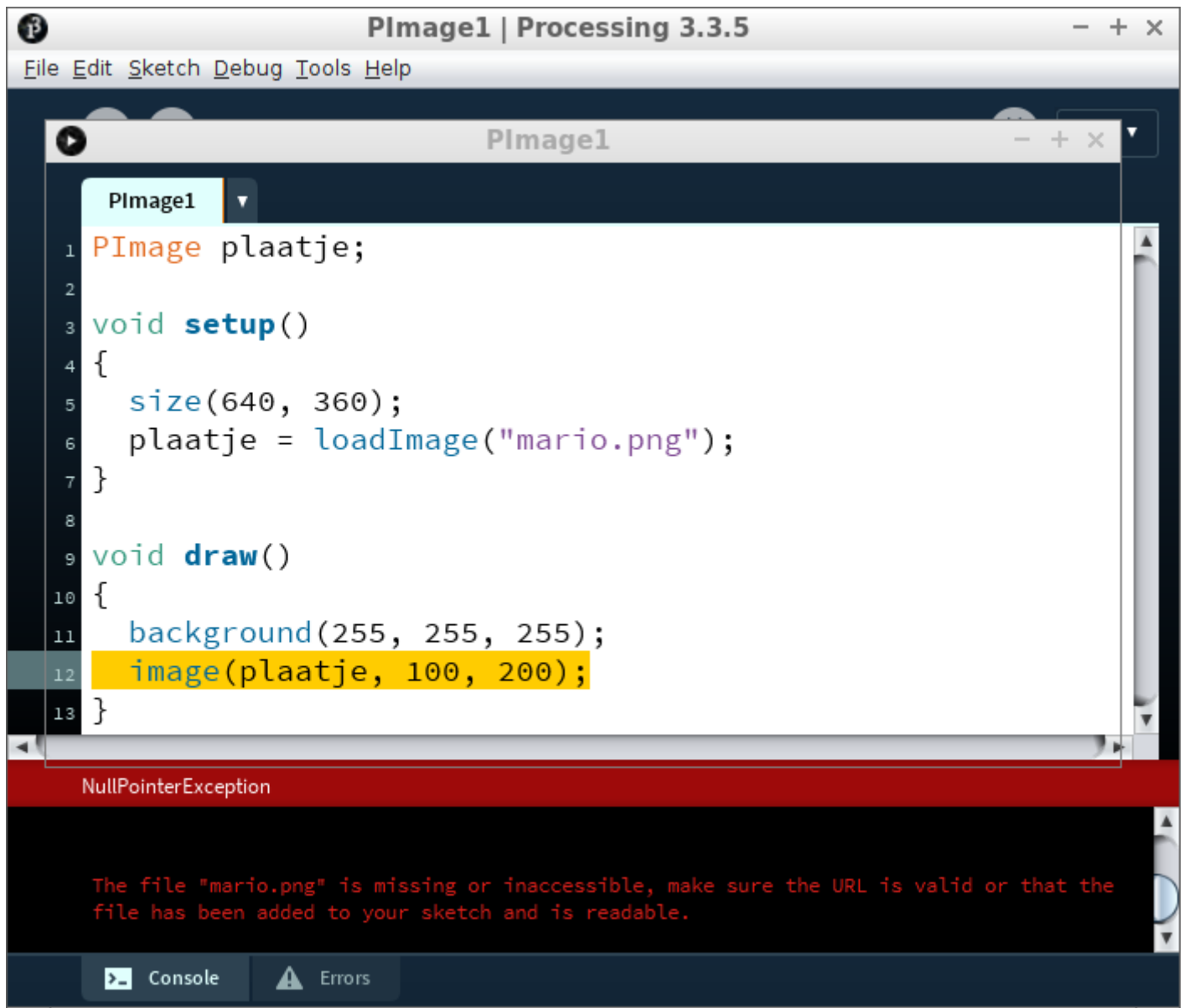


Figure 9: PImage1.png

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!

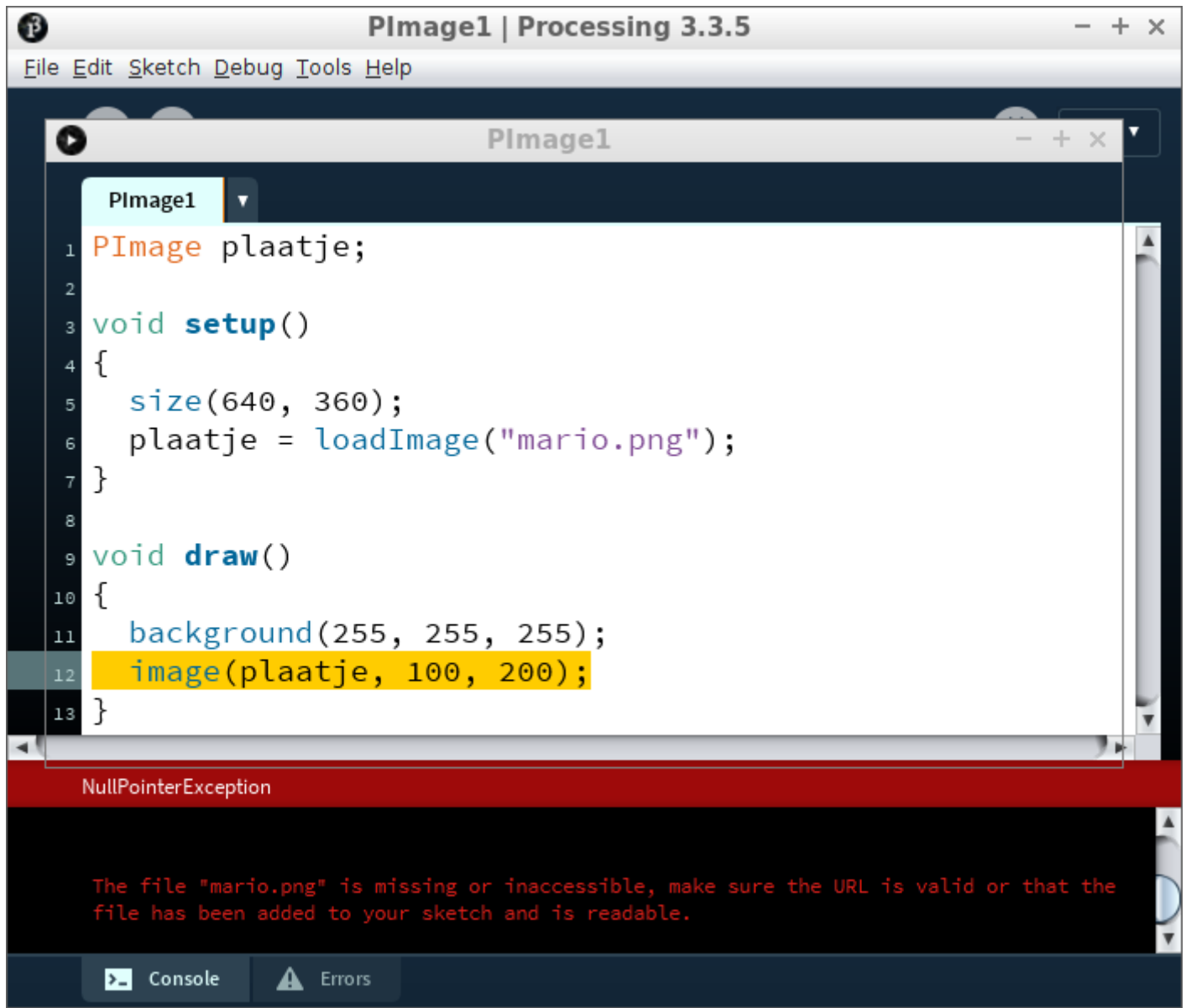


Figure 10: Oplossing 1



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:

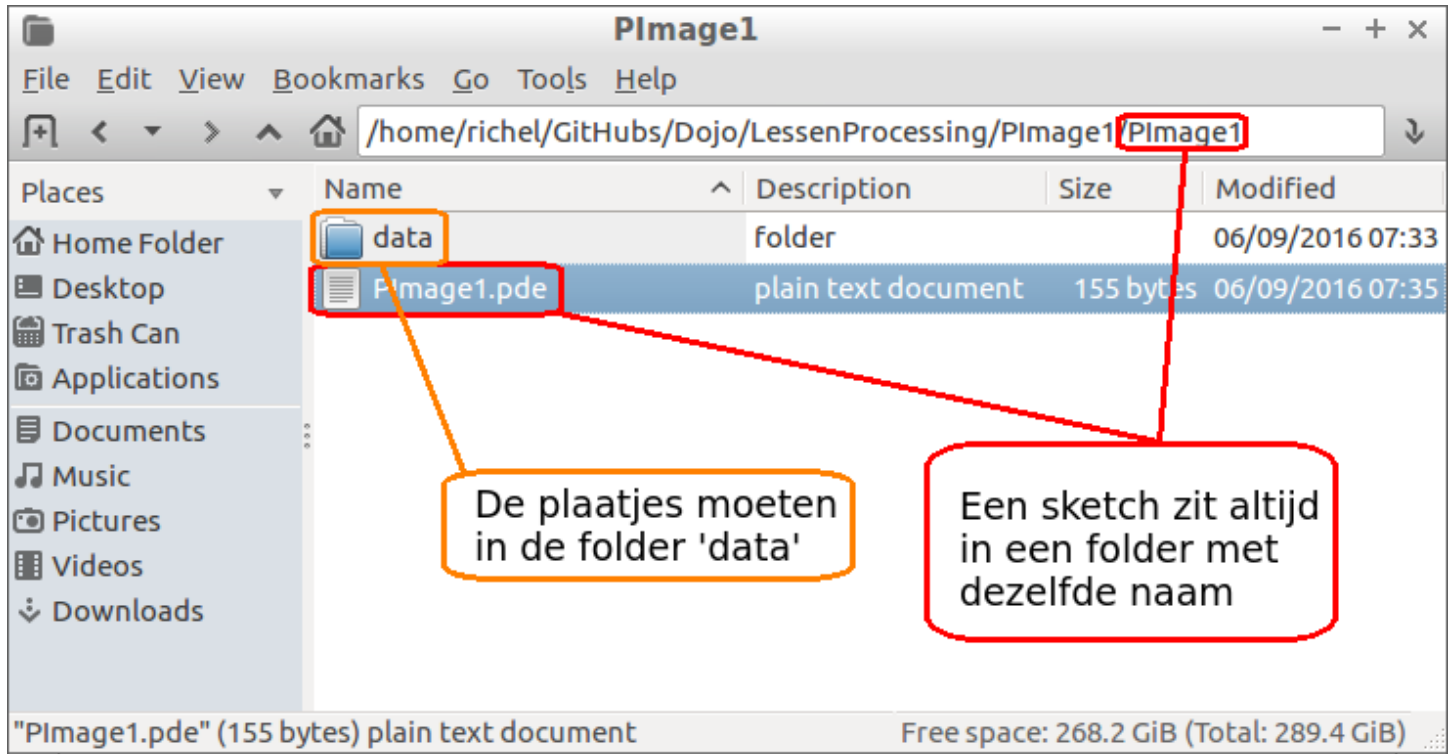


Figure 12: Folder structuur

- 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.

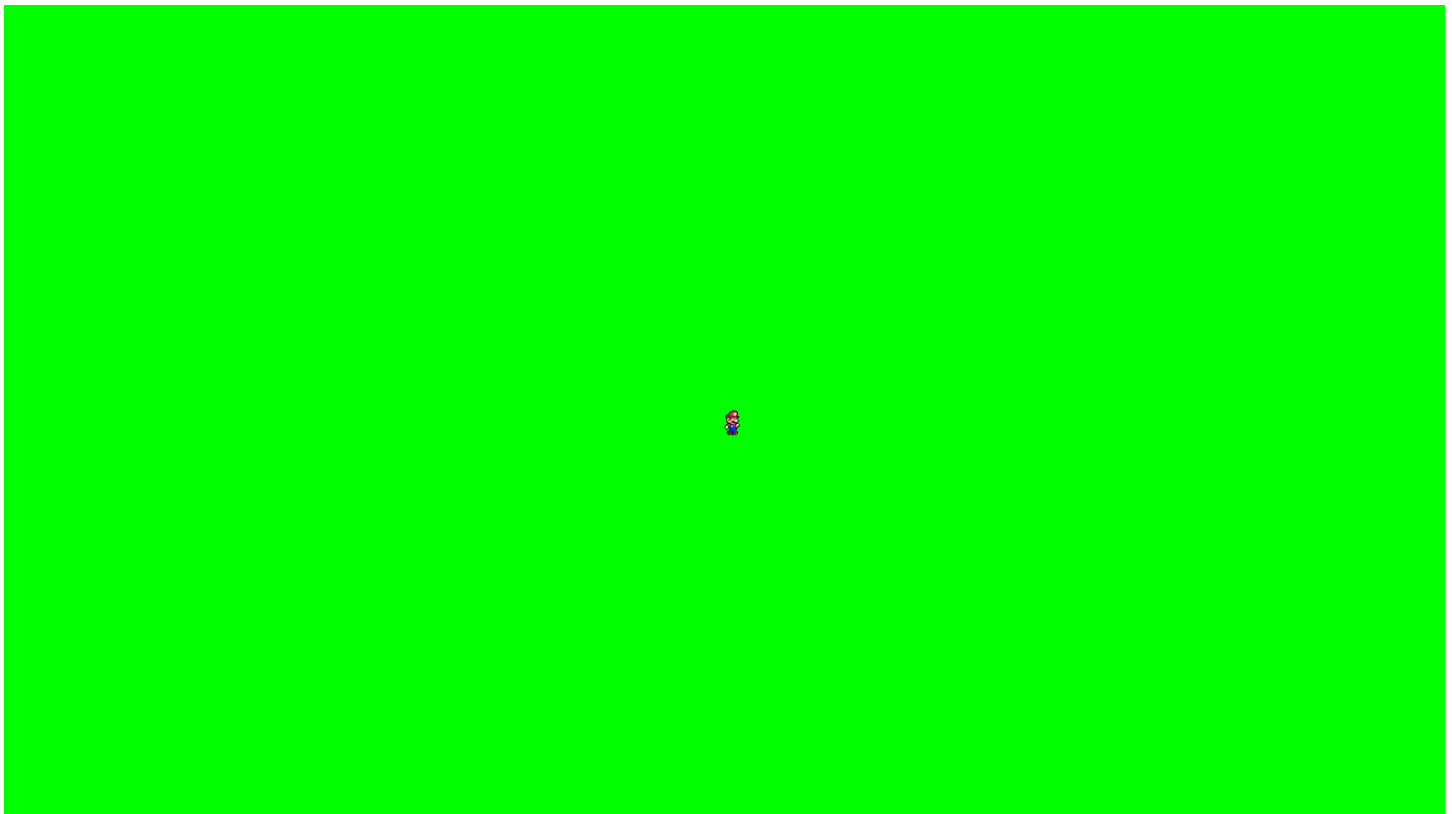


Figure 13: PImage eindopdracht

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;
  }
}
```

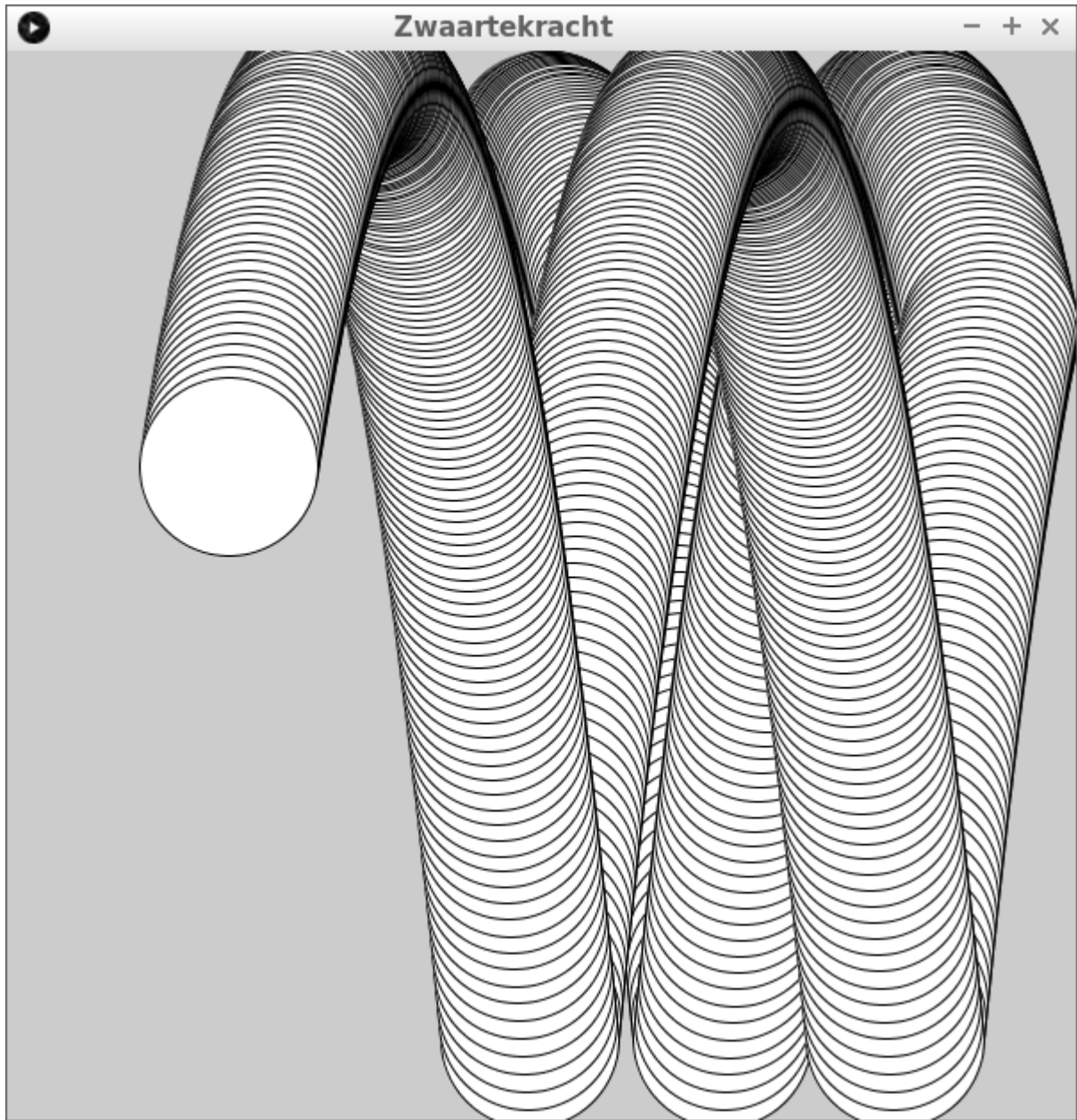


Figure 14: Zwaartekracht

```

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

```

Oplossing 1

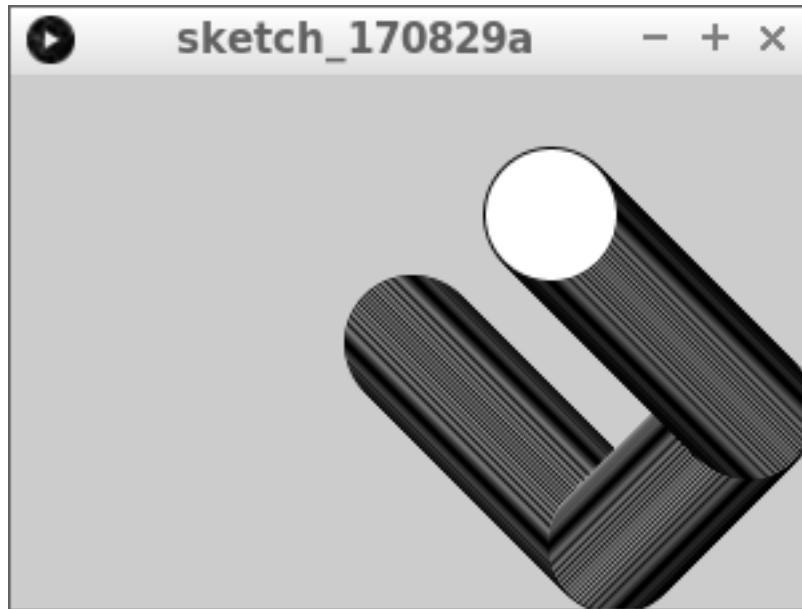


Figure 15: Zwaartekracht oplossing 1

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;
}

```

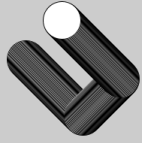


Figure 16: Zwaartekracht opdracht 2

```
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;
  }
}
```

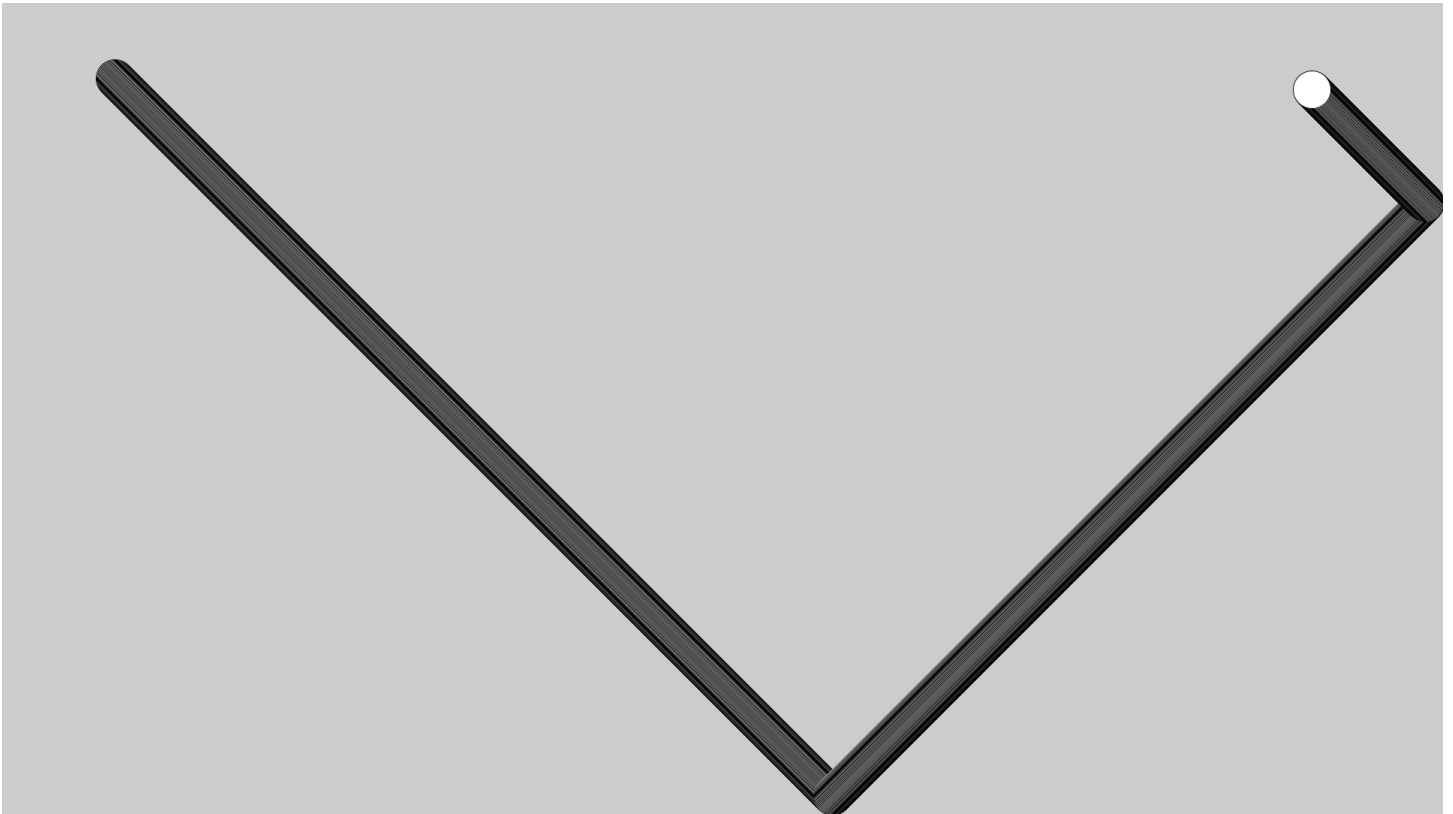


Figure 18: Zwaartekracht opdracht 4

```
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!

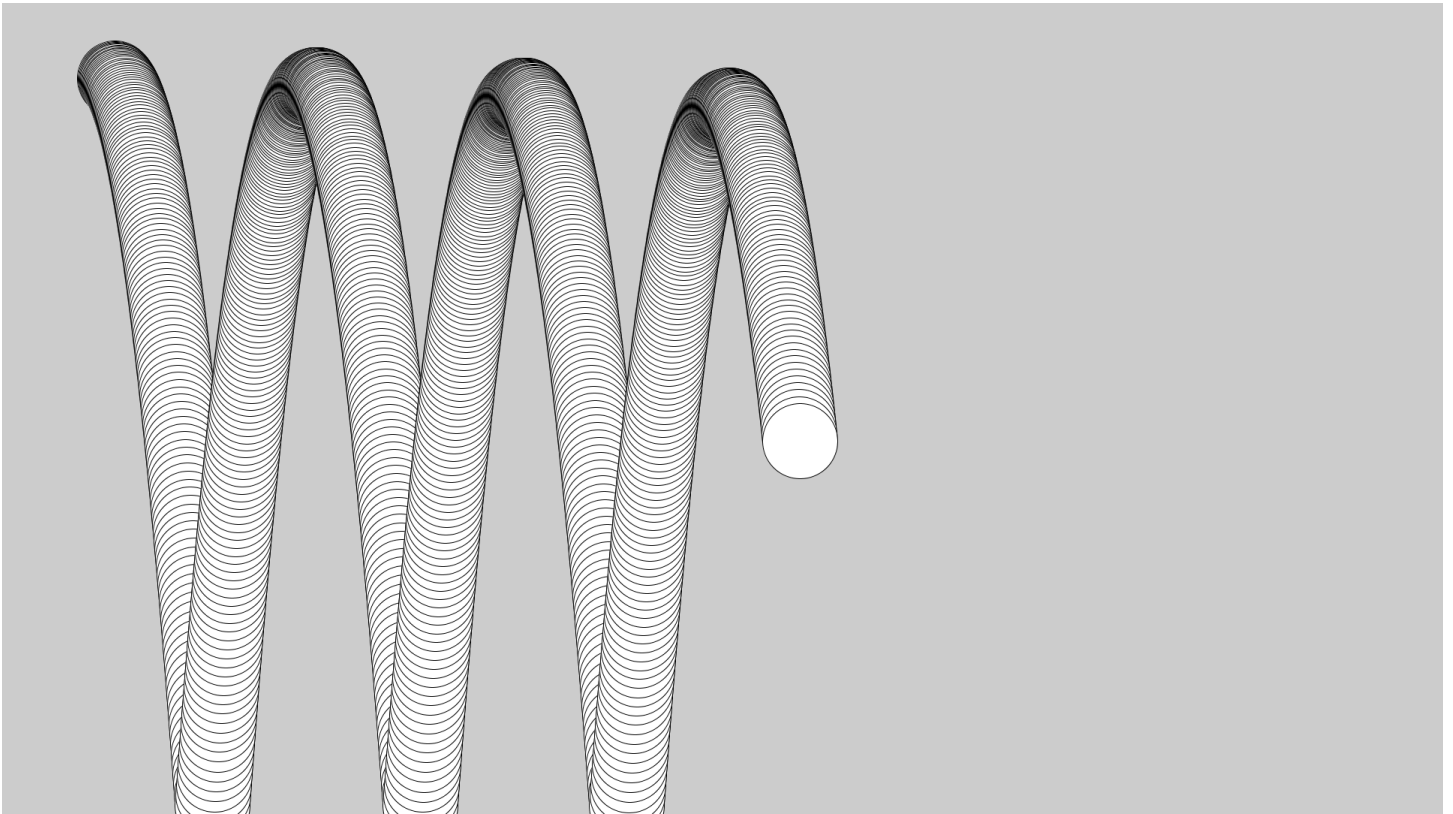


Figure 19: Zwaartekracht eindopdracht

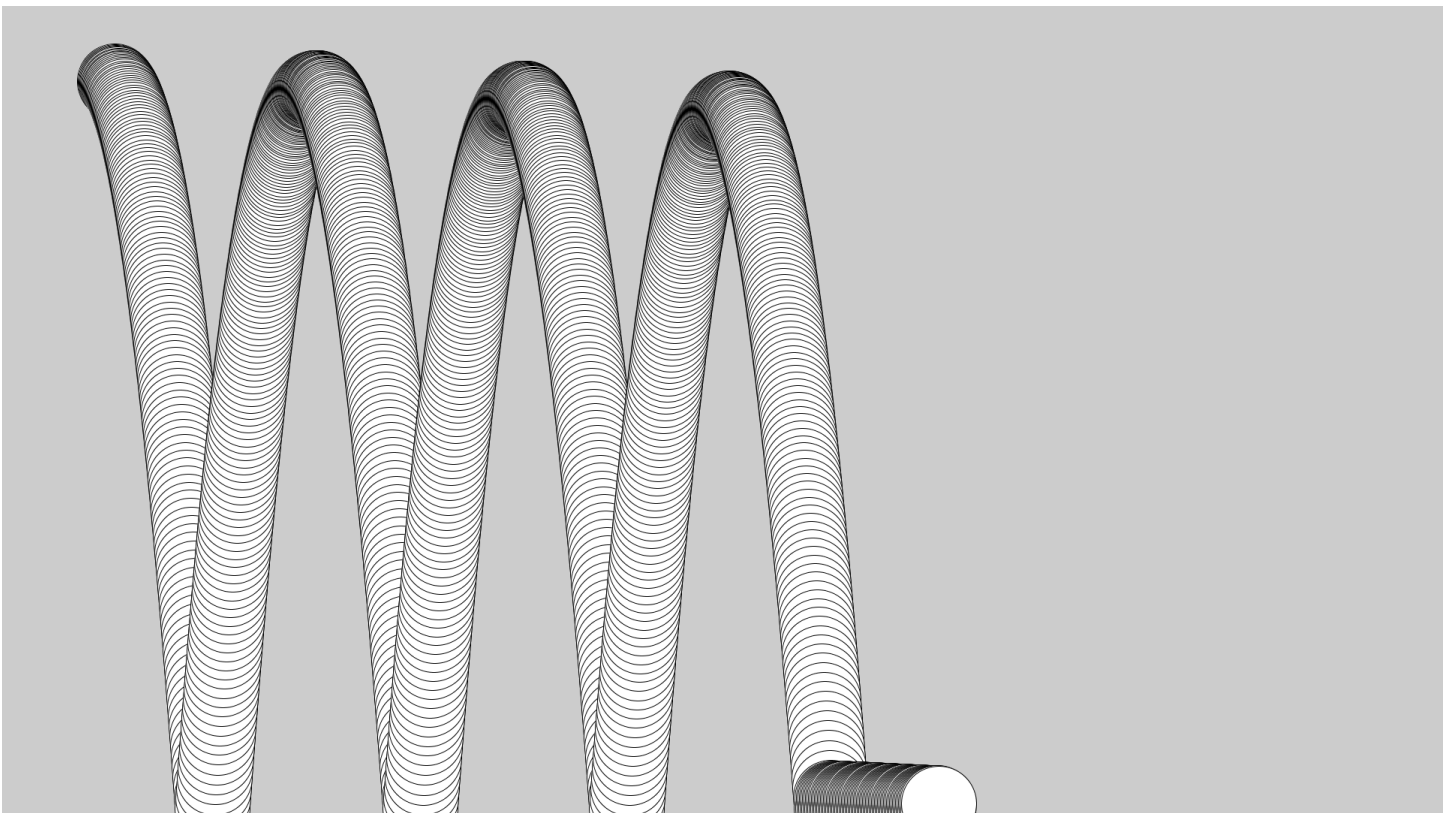


Figure 20: Zwaartekracht eindopdracht ook goed

Arrays1

Met arrays kun je de computer veel waardes laten onthouden: de coördinaten 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:

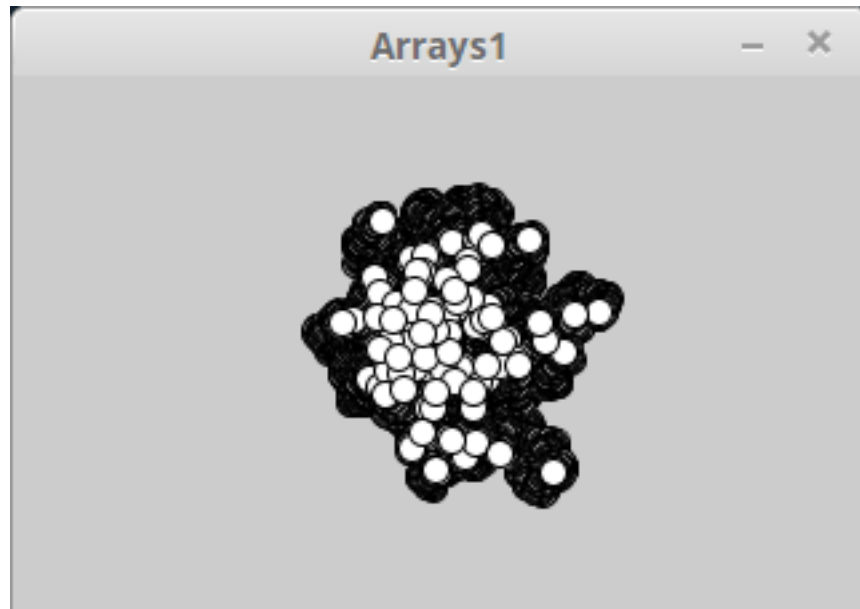


Figure 21: Honderd rookdeeltjes

Twee ballen die eeuwig naar rechts gaan

Je hebt een bal eeuwig horizontaal laten bewegen:

```
float x = 0;

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

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

Vragen

- 1. Run deze code. Wat zie je?
- 2. Zorg dat er een tweede bal bijkomt. Tips:

- verander de naam `x` naar `x1`
- maak een nieuwe variabele met de naam `x2`
- 3. Hoeveel regels code kost het ongeveer om een extra bal te programmeren?

Oplossing



Figure 22: Twee ballen die eeuwig naar rechts gaan

- 1. Een bal die horizontaal heen en weer beweegt
- 2. Zo doe je dat:

```
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;
  }
}
```

- 3. zeven regels

Vragen: drie ballen

- 1. Zorg dat er een derde bal bijkomt
- 2. Hoeveel regels code kost het ongeveer om een extra bal te programmeren?
- 3. Hoeveel extra regels kost het om nog eens tien ballen erbij te programmeren?

Oplossing

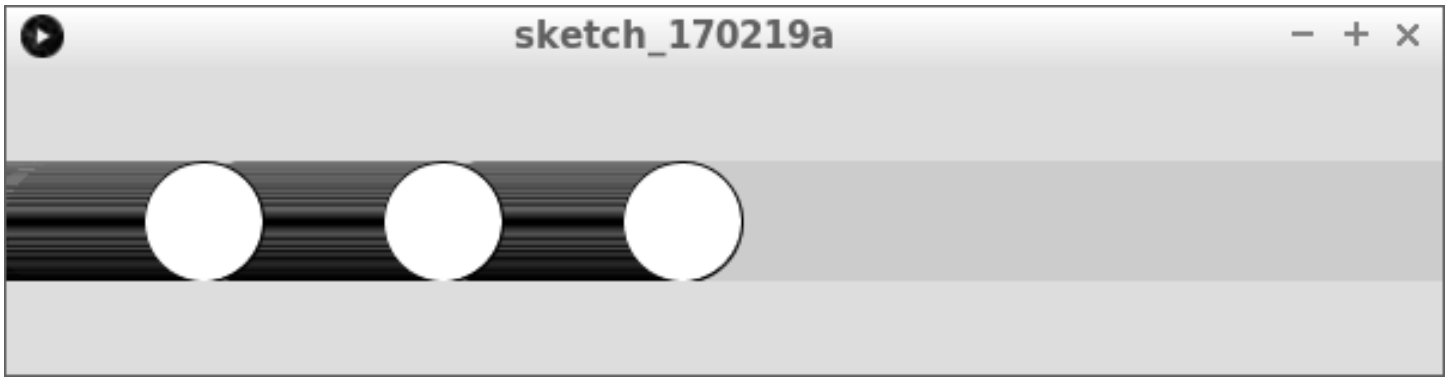


Figure 23: Drie ballen

- 1. Zo doe je dat:

```
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;
  }
}
```

- 3. weer zeven regels
- 4. tien keer zeven is zeventig regels

Waarom arrays?

Je hebt de code gezien met de drie ballen die eeuwig naar rechts gaan.

Het valt op dat er veel herhaling in zit. Dit komt omdat we de computer steeds een getal tegelijk laten onthouden: `float x1 = 300` betekent: 'Lieve computer, onthoudt een gebroken getal met de naam `x1`, met als beginwaarde 300'. Wat we willen kunnen zeggen is 'Lieve computer, onthoud keiveel gebroken getallen'. Dit is precies wat een array kan doen!

Wat is een array?

Een array kun je zien als een kast met laatjes. In deze les beginnen we met een kast met een laatje:



Figure 24: Kast met laatje

Elk laatje heeft een nummer en in elk laatje kan een getal.

Hier zie je het nummer van het laatje, en het getal wat erin zit:

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

Het valt op dat het laatje nummer *nul* heeft. Je zegt: ‘Het eerste laatje heeft index nul’. Als je normaal telt, begin je bij een. Bij indices (het meervoud van index) begin je te tellen bij nul. De kast heeft een laatje, met index nul.

Vragen

- Wat is een array?

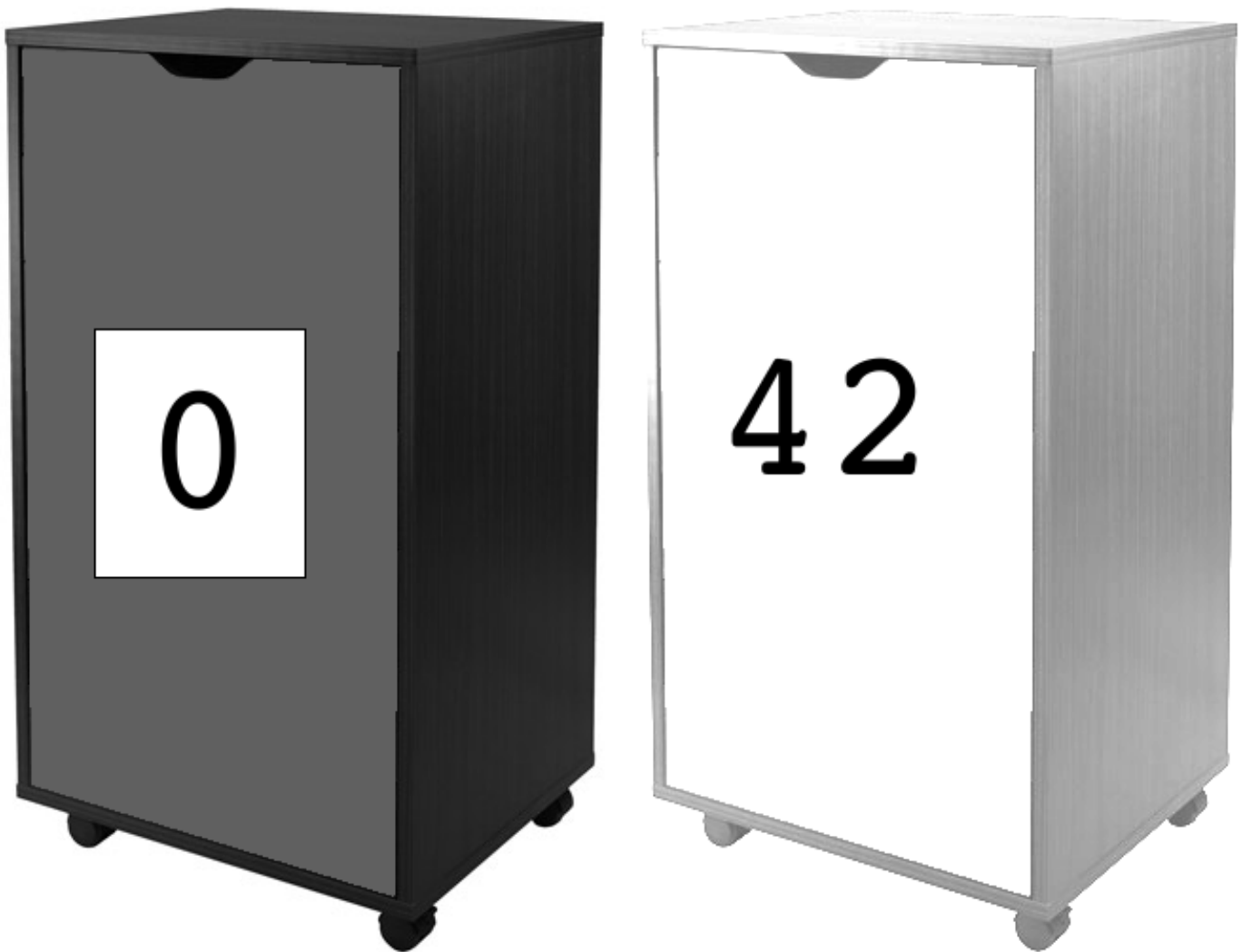


Figure 25: Kast met genummerde laatjes

- Wat is een index?
- Wat is de laagste index?

Werken met een array met een laatje

geheime_getallen

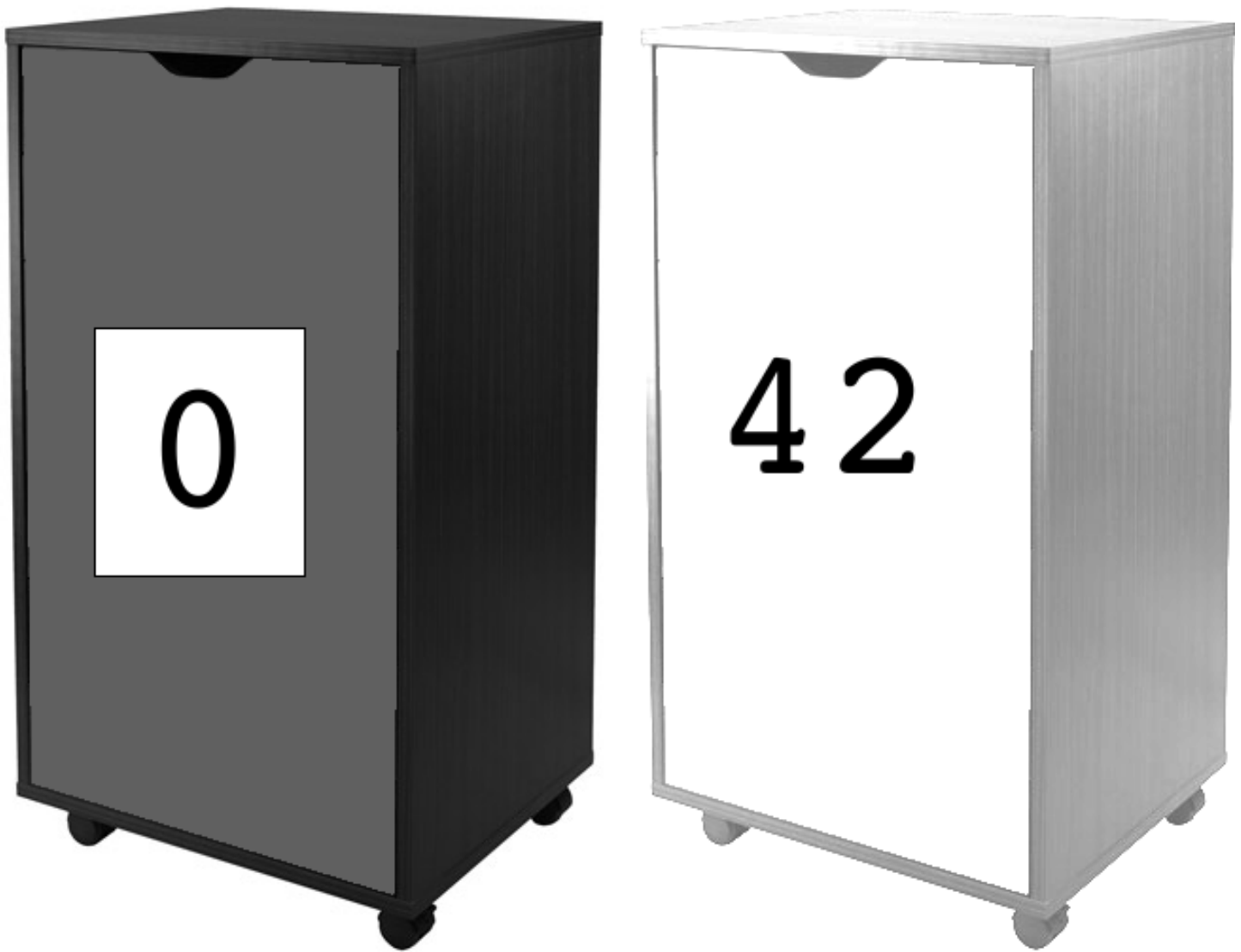


Figure 26: Array met naam 'geheime_getallen' en een laatje

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;
```

Hiermee zeg je: '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 zeg je: 'Lieve computer, maak `geheime_getallen` groot genoeg om een gebroken getal (floats) te onthouden'.

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

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

Hiermee zeg je, in de derde regel: ‘Lieve computer, stop in laatje met index nul het getal tweeënveertig’. Deze code zou prima in de `setup` functie gedaan kunnen worden.

Je zou ook de waarde in de laatjes kunnen lezen:

```
float x = geheime_getallen[0];  
ellipse(x,200,300,400);
```

Hiermee zeg je: ‘Lieve computer, kijk wat er in laatje met index nul zit en onthoud dat als `x`. Teken dan een ovaal die `x` pixels naar rechts is, 200 pixels omlaag is, 300 pixels breed is, en 400 pixels hoog is.’

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.

Vragen

- Welke foutmelding krijg je als je `float[] geheime_getallen;` in de `setup` functie zet?
- Welke foutmelding krijg je als je `float geheime_getallen;` (dus zonder blokhaken) gebruikt?
- Je wilt een array maken van gebroken getallen met de naam `snelheden`. Hoe zeg je dat in code?

Drie ballen met een array

We hebben al de code van drie ballen die eeuwig naar rechts gaan. Deze gaan we nu in een array stoppen

Opdracht

- 1. Verander de code van ‘Drie ballen die eeuwig naar rechts gaan’ met een array:
- Het programma moet precies hetzelfde doen
- Gebruik in plaats van de variabelen `x1`, `x2` en `x3` de naam `xs` (meerdere `x`-en)
- De array `xs` wordt dus drie groot
- 2. Hoeveel regels wordt je code korter?

Oplossing

- 1. Zo doe je dat:

XS

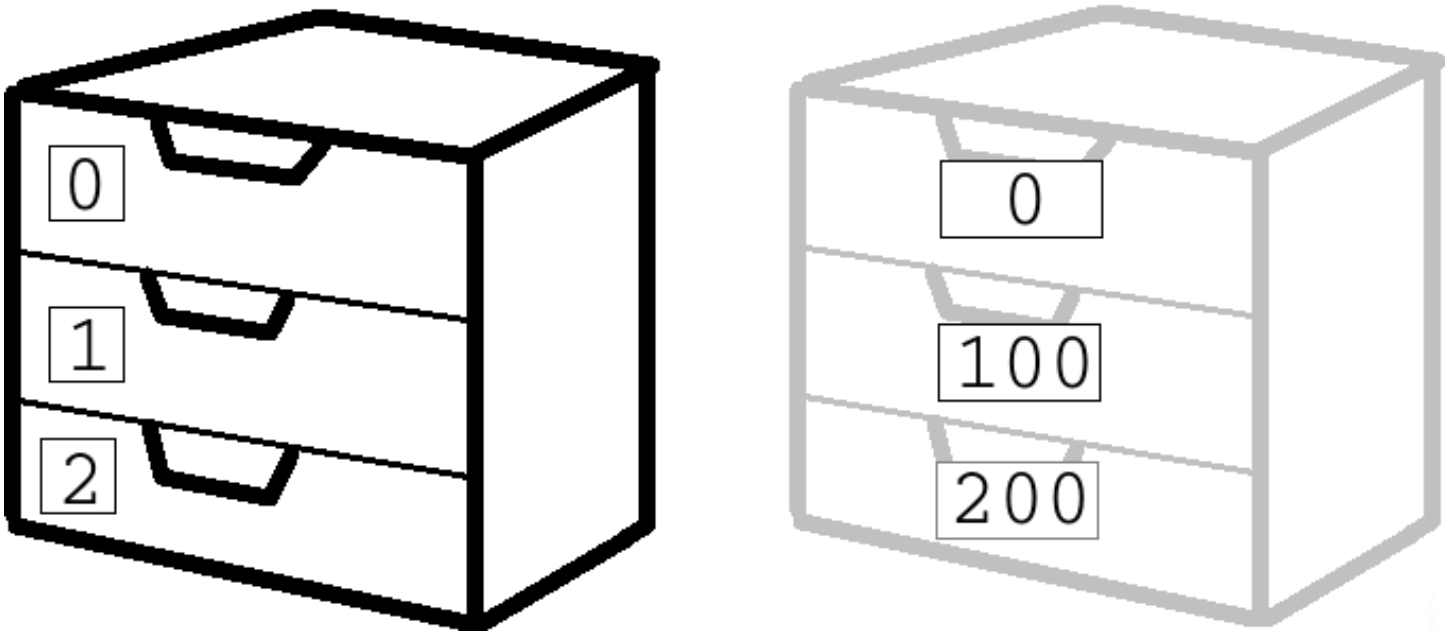


Figure 27: Array voor ‘Drie ballen met een array’

```
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;
  }
}
```



```
}  
}
```

- 2. Je code wordt juist langer!

Opdracht

- 1. Maak de code korter door for-loops te gebruiken
- 2. Wordt de code nu langer of korter?

Oplossing

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

Vier ballen

Om nu een vier bal erbij te maken, pak je de code van ‘Drie ballen’ en maakt van de 3 een 4

Opdracht

- 1. Pak de code van ‘Drie ballen’ en maakt van de 3 een 4
- 2. Op hoeveel plekken moest je een 3 in een 4 veranderen?

Oplossing

- 1. Dat ziet er zo uit:

```
float[] xs;  
  
void setup()  
{  
    size(600, 50);  
    xs = new float[4];  
}
```

XS

0	
1	
2	
3	

0
100
200
300

Figure 28: Array met vier laatjes

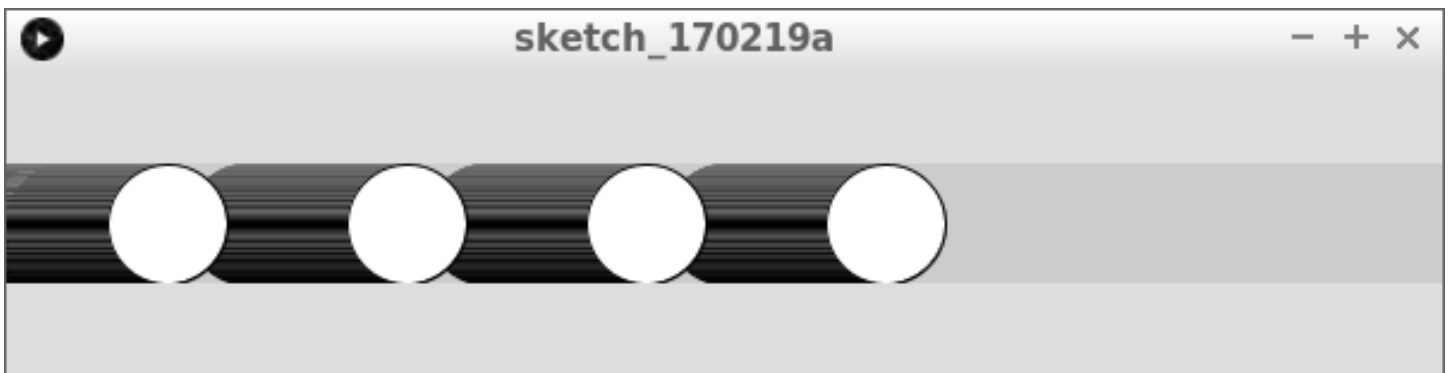


Figure 29: Vier ballen

```

    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;
        }
    }
}

```

- 2. Op drie plekken

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 coördinaten 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:

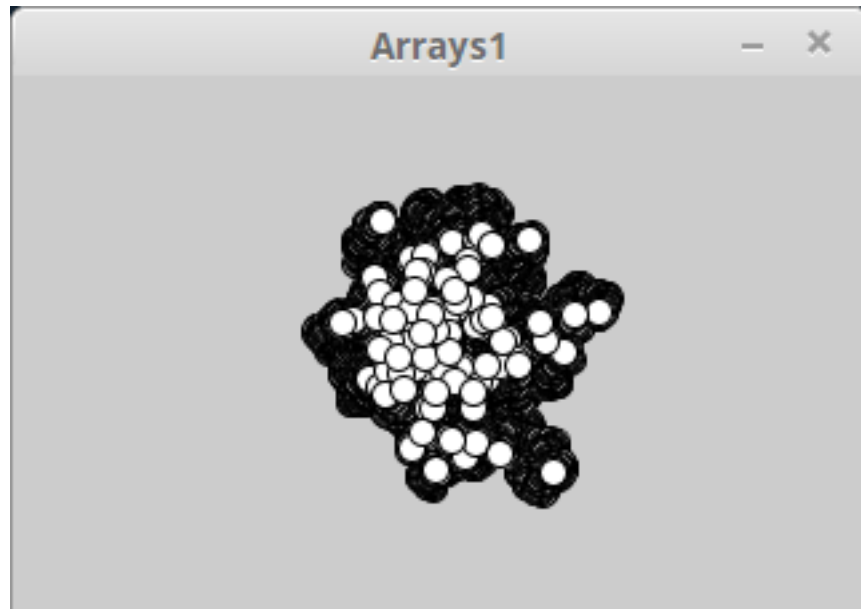


Figure 30: Honderd rookdeeltjes

Een rookdeeltje

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



Figure 31: 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` coördinaat 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` coördinaat 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);
}
```

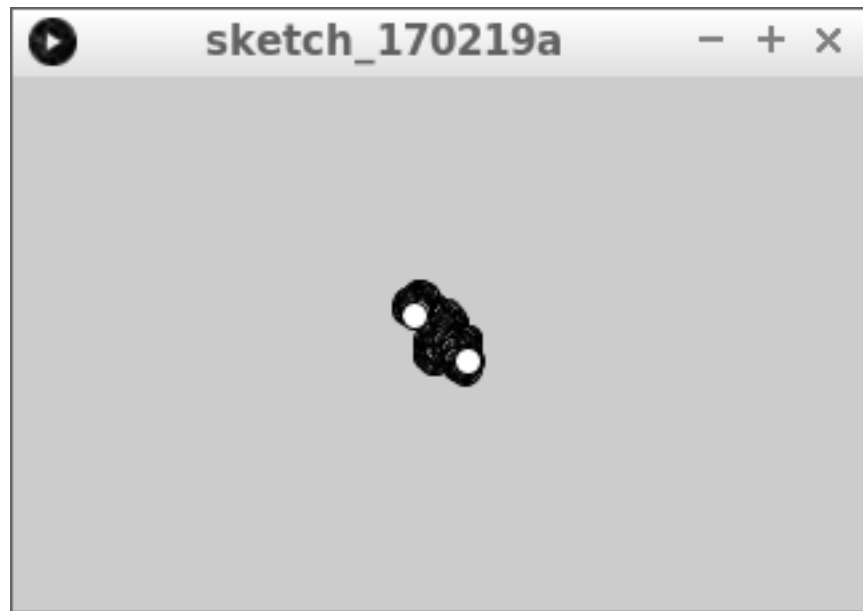


Figure 32: Twee rookdeeltjes

```
}
```

- 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;
}
```

XS

0	
1	

160
160

ys

0	
1	

100
100

Figure 33: 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 34: 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 35: 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()
{

```

xs

0	160
1	160
2	160
3	160

ys

0	100
1	100
2	100
3	100

rs

0	0
1	64
2	128
3	196

Figure 36: 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

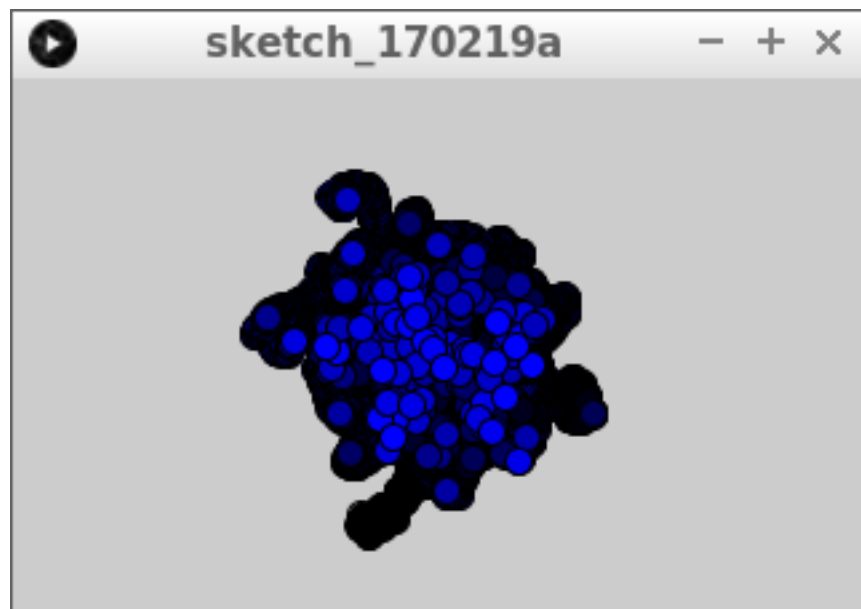


Figure 37: Arrays2 eindopdracht

Maak nu de code zo dat:

- er 256 rookdeeltjes komen.
- elk rookdeeltje heeft een 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**)