

Figure 1: Boek 4

Contents

Voorwoord	1
fullScreen	2
PImage	14
Zwaartekracht	19
Arrays 1	29
Arrays2	48

Voorwoord



Figure 1: Het logo van De Jonge Onderzoekers

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 2: De licentie van dit boek

(C) Dojo Groningen 2016-2018

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.

fullScreen: 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’

fullScreen: oplossing 1

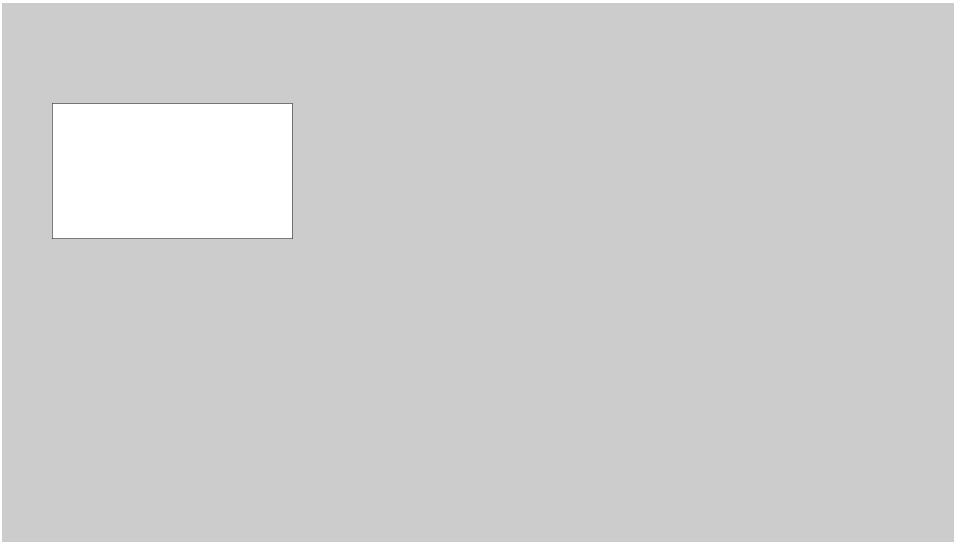


Figure 3: **fullScreen:** oplossing 1

fullScreen: opdracht 2

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

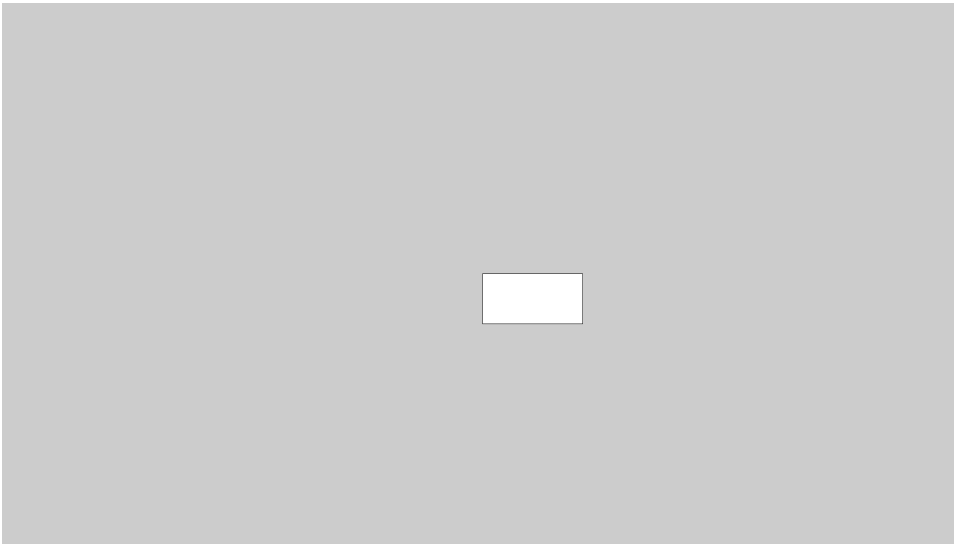


Figure 4: **fullScreen:** opdracht 2

fullScreen: oplossing 2

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

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

fullScreen: opdracht 3

Zet nu de rechthoek in het midden van het scherm.

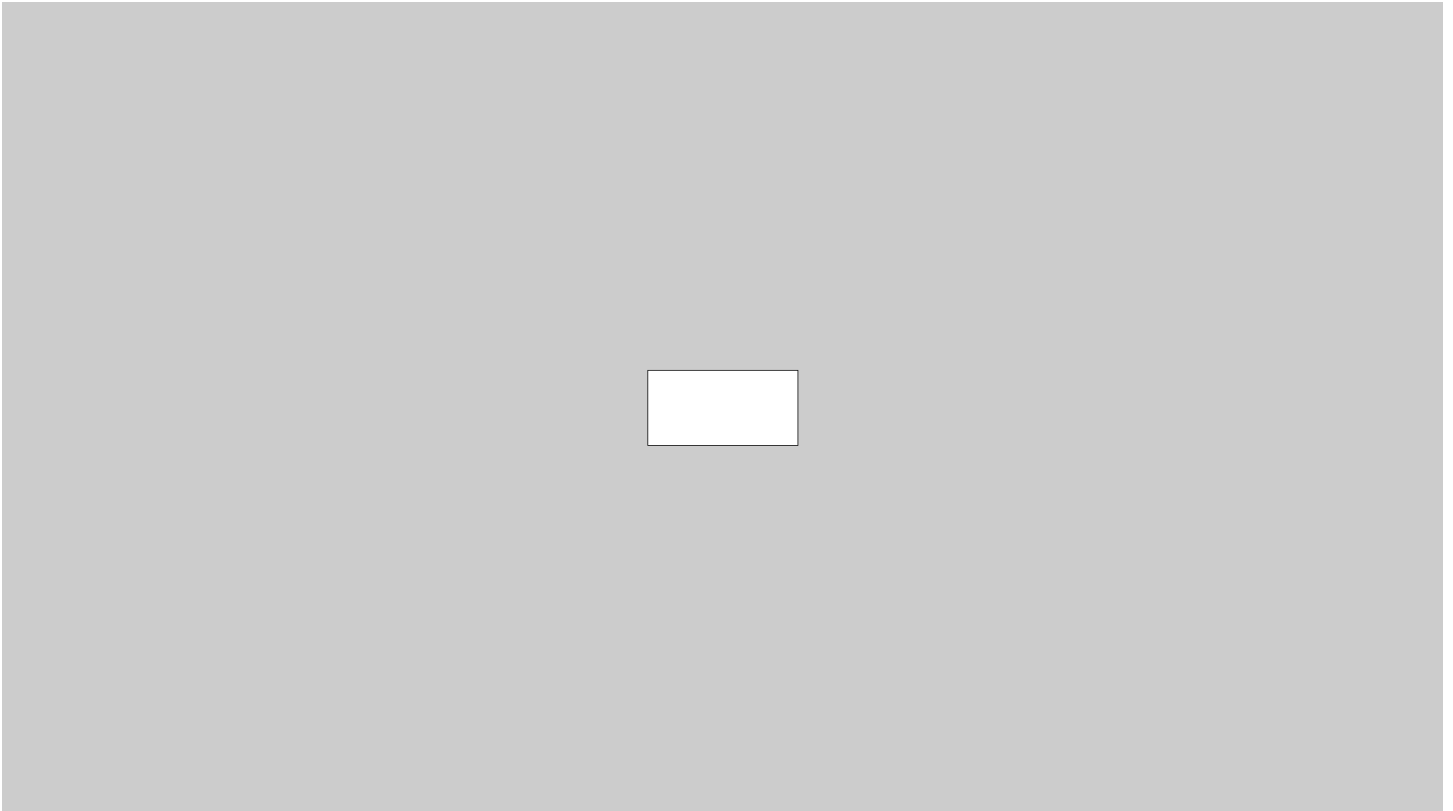


Figure 5: fullScreen: opdracht 3



De rechthoek moet 100 naar links en 50 omhoog

fullScreen: oplossing 3

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

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

fullScreen: opdracht 4

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

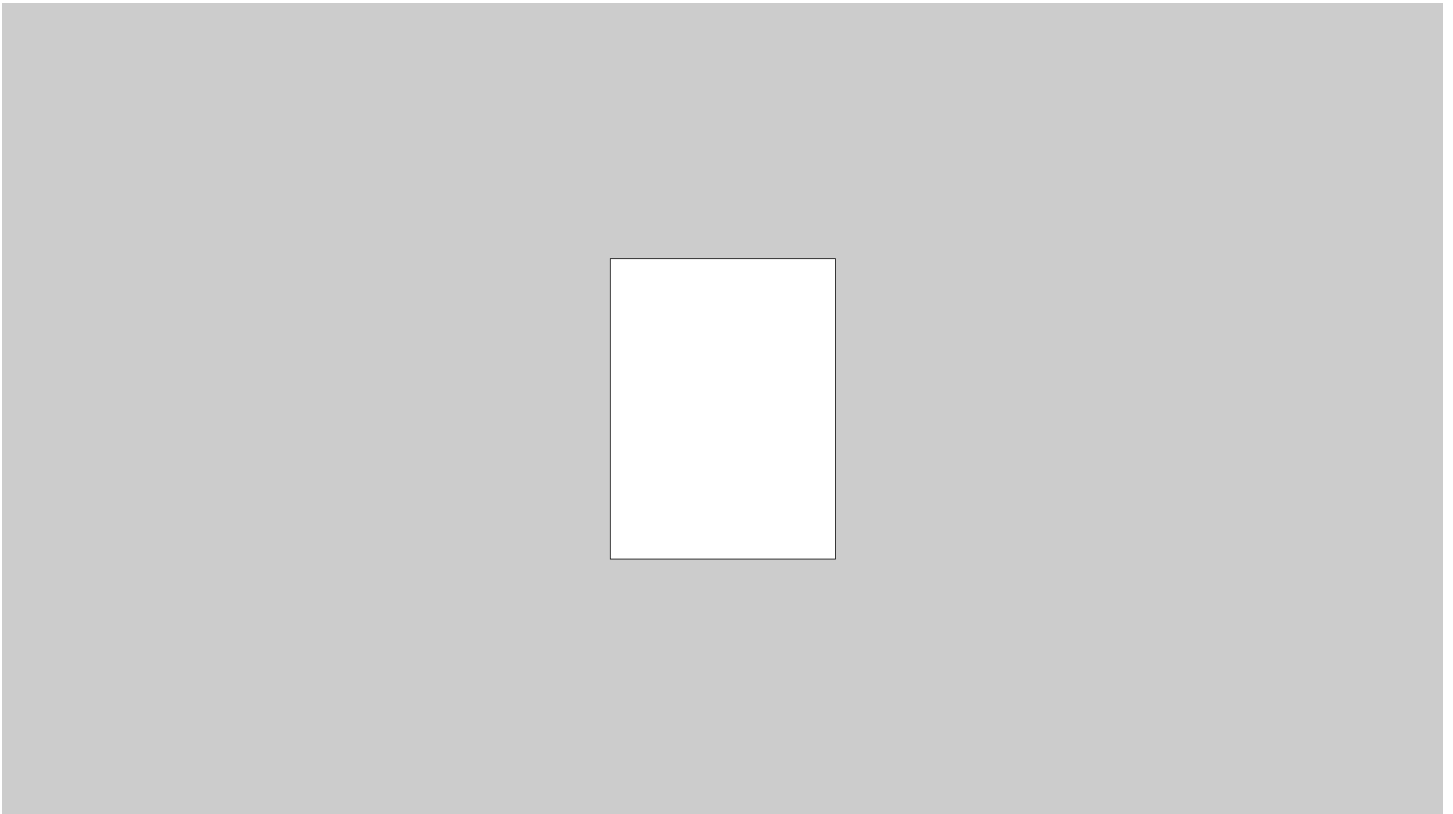


Figure 6: **fullScreen:** opdracht 4

fullScreen: oplossing 4

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

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

fullScreen: opdracht 5

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

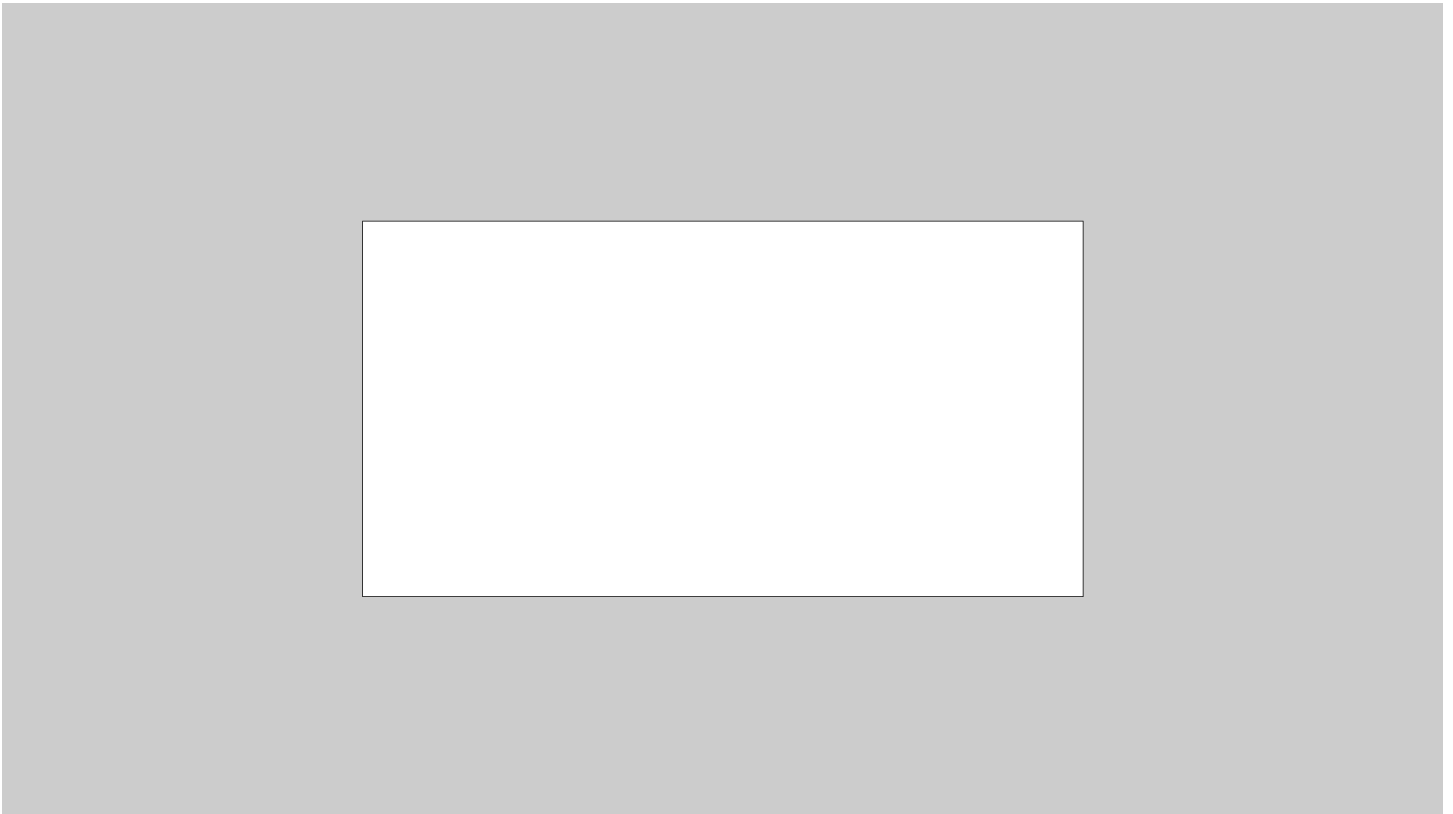


Figure 7: **fullScreen:** opdracht 5

fullScreen: oplossing 5

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

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

fullScreen: eindopdracht

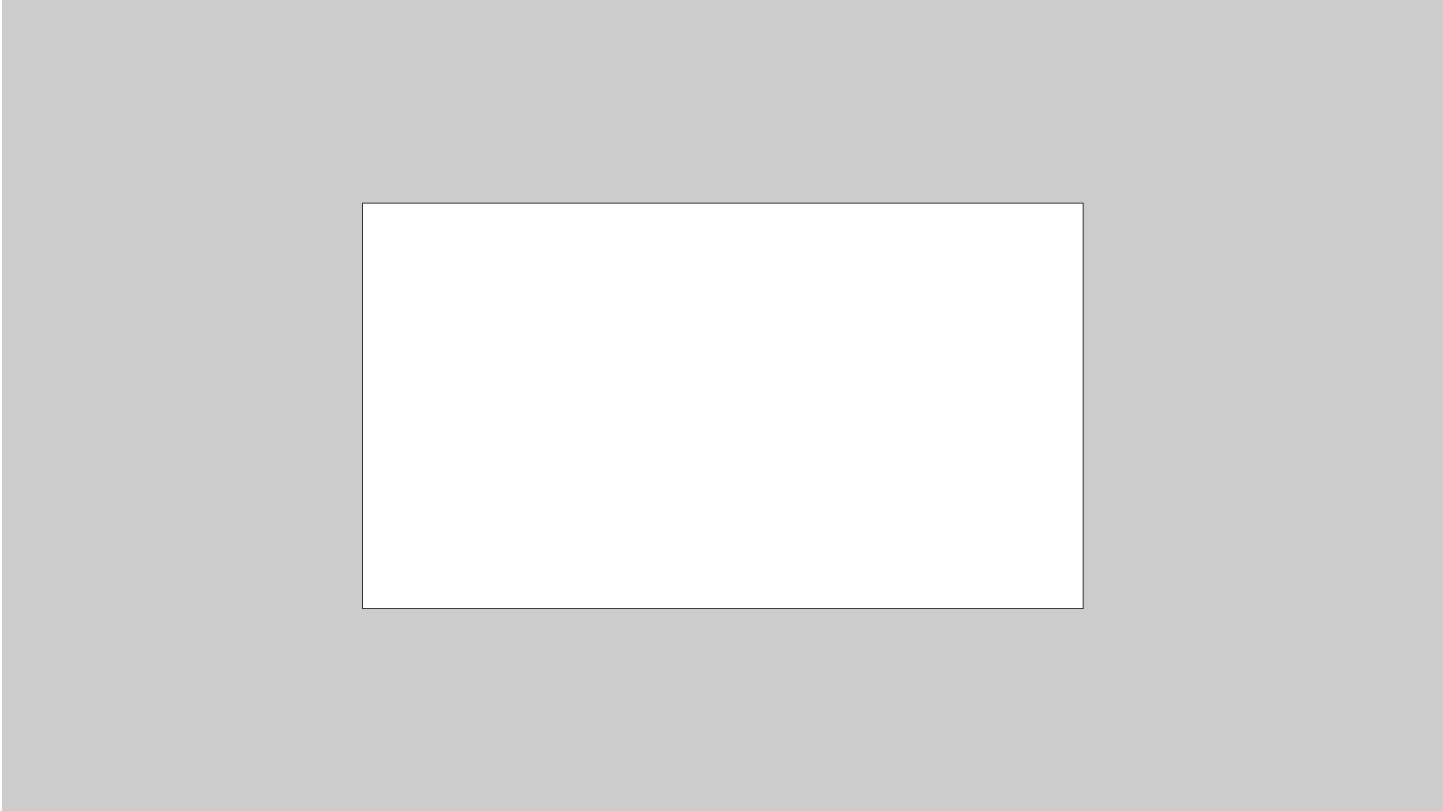


Figure 8: **fullScreen:** eindopdracht

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: EmojiSunglasses.png

PImage: 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);
}
```


PImage: oplossing 1

Je krijgt een error!

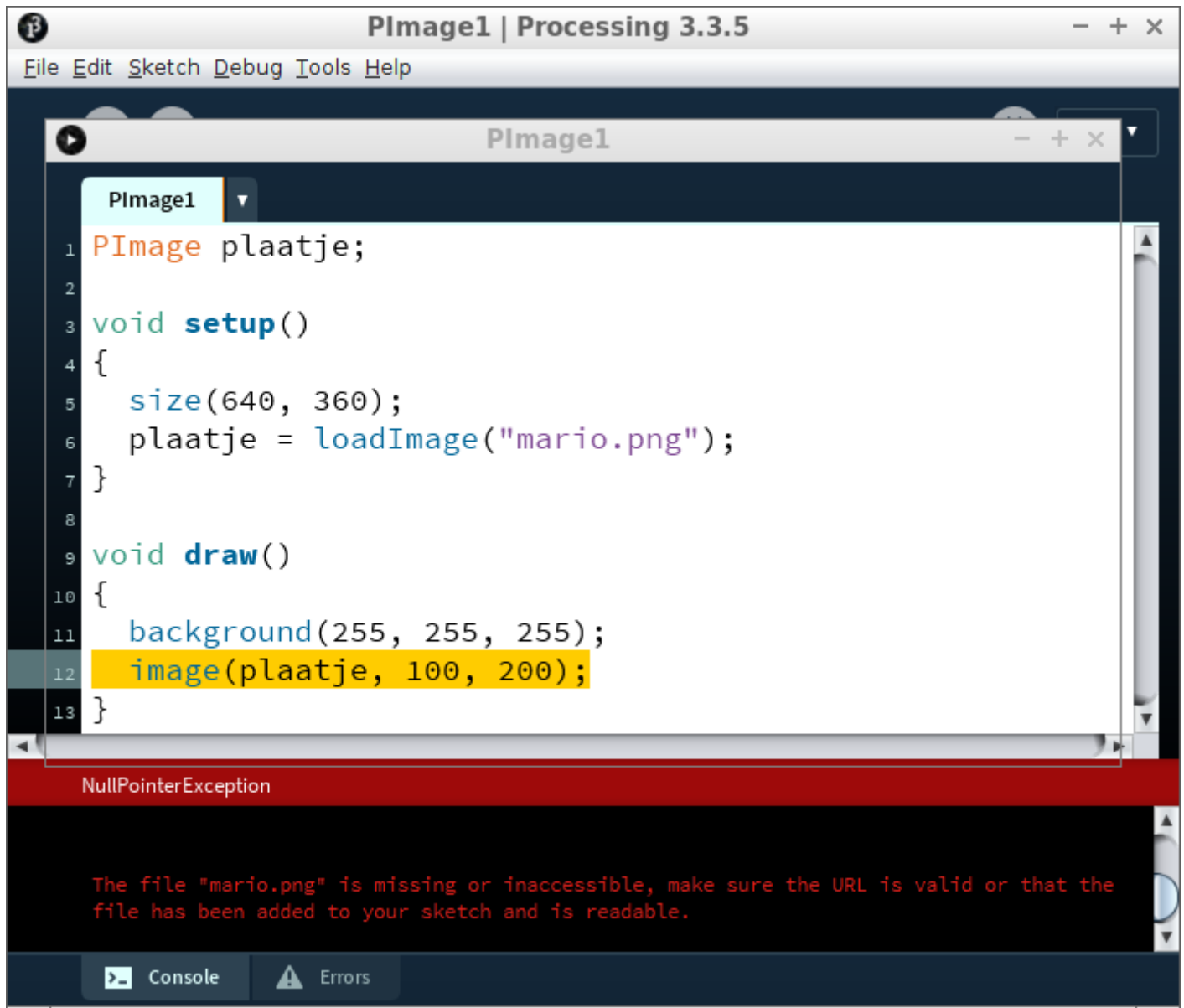


Figure 10: Oplossing 1



De computer zegt dat hij het plaatje niet kan vinden!

PImage: 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:

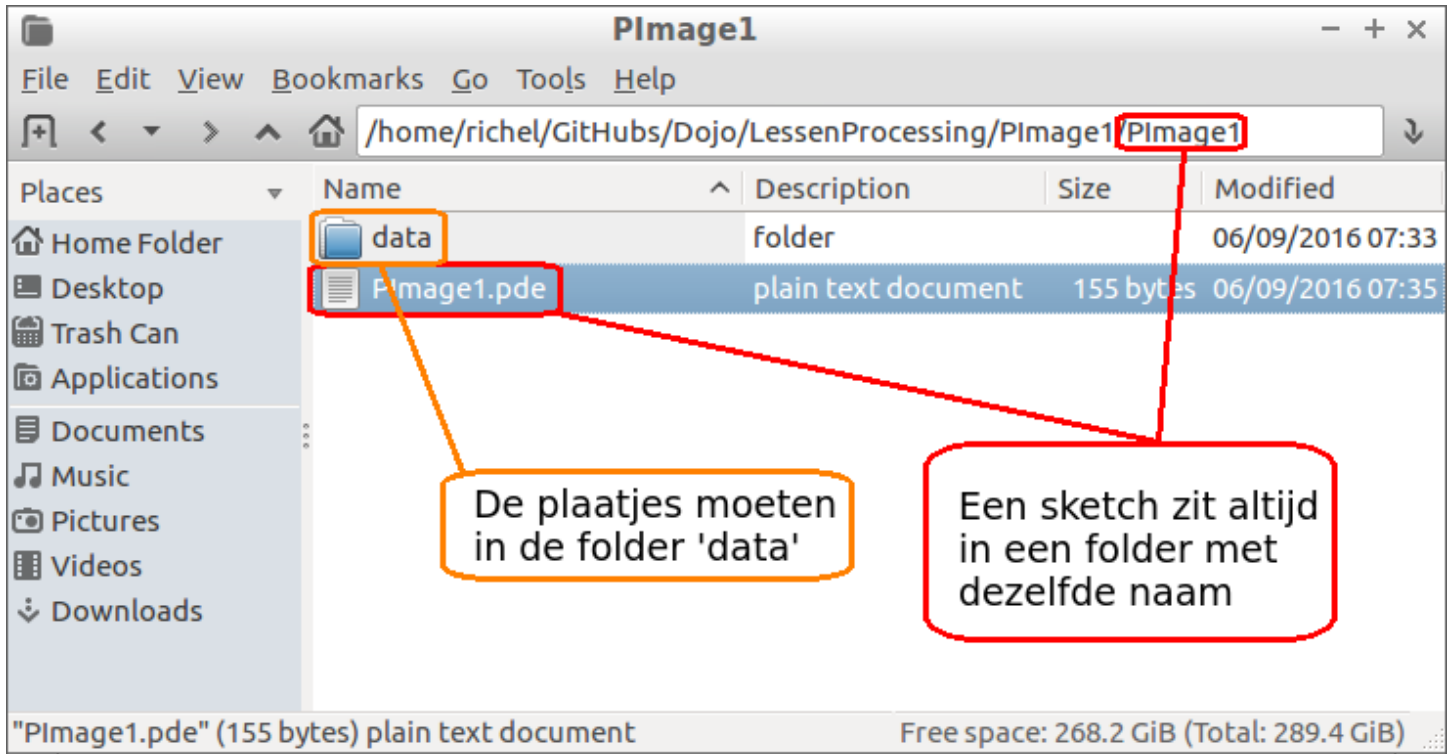


Figure 12: Folder structuur

- 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`

PImage: eindopdracht

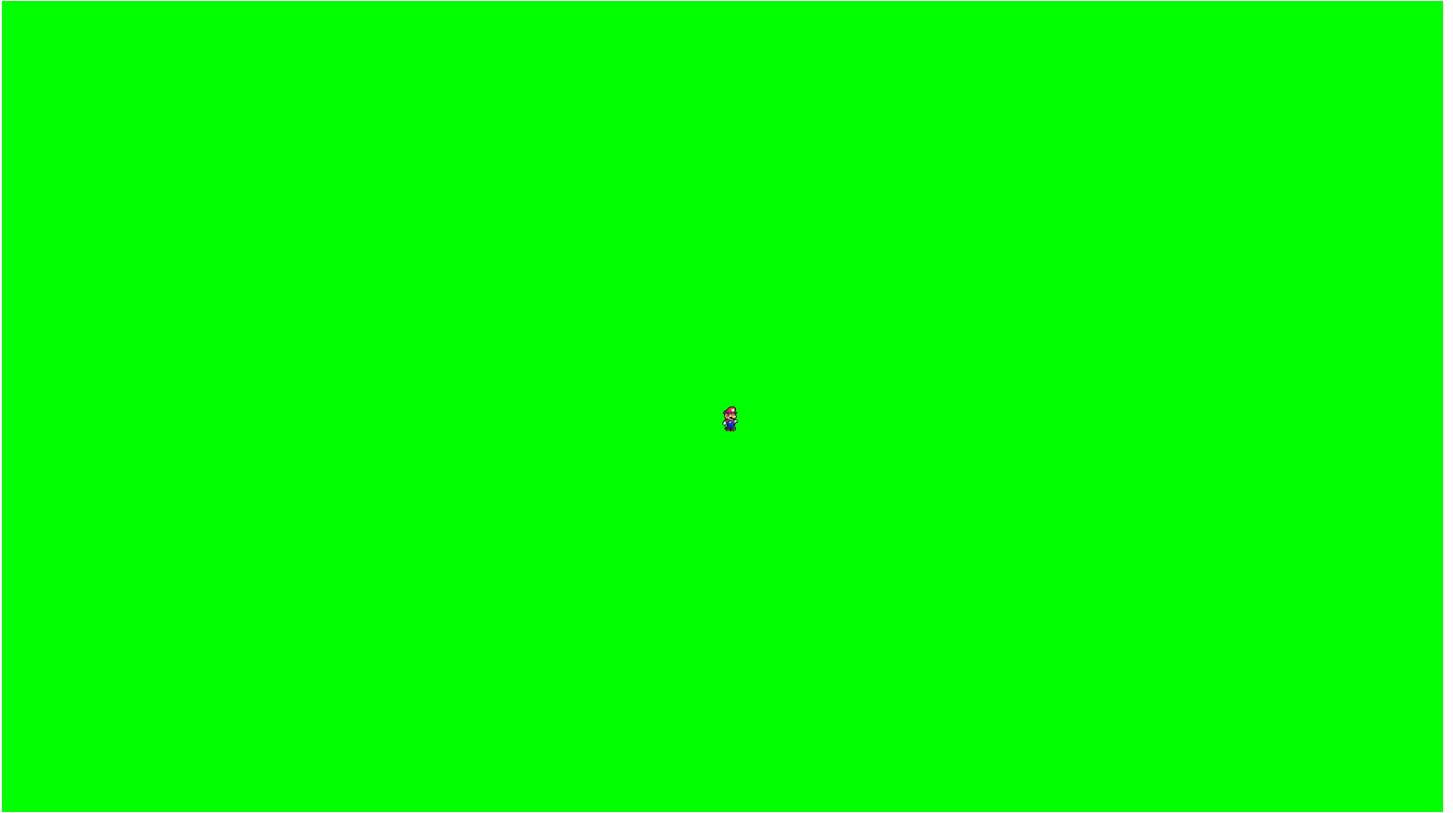


Figure 13: PImage: eindopdracht

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

Zwaartekracht

In deze les gaan we zwaartekracht programmeren.

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

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

Zwaartekracht: oplossing 1

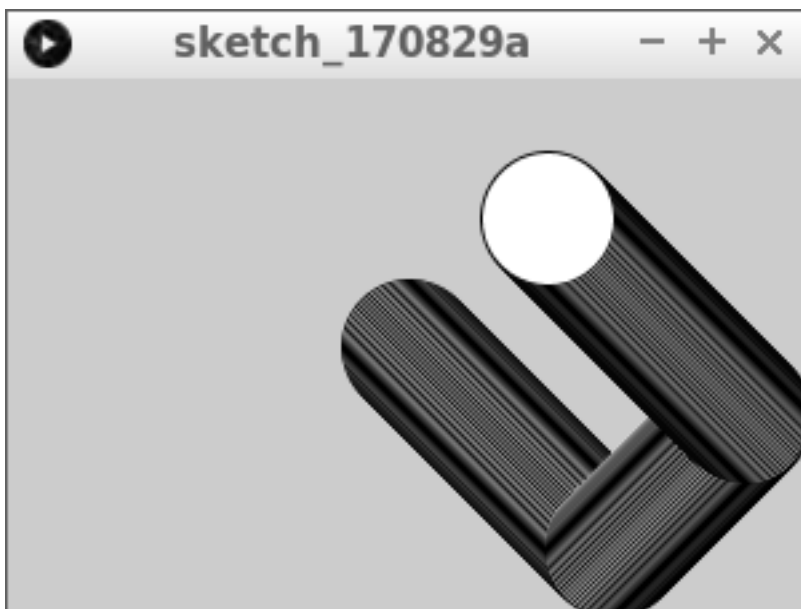


Figure 14: Zwaartekracht: oplossing 1

Zwaartekracht: opdracht 2

Zorg dat het programma full-screen wordt



Figure 15: Zwaartekracht: opdracht 2

Zwaartekracht: 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;
  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;
  }
}
```


Zwaartekracht: opdracht 3

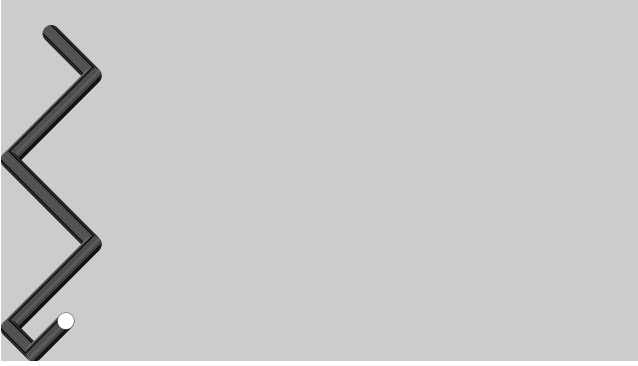


Figure 16: Zwaartekracht: opdracht 3

Zorg dat de bal goed aan de onderkant stuitert.



Tip: vervang 175 door `height - 25`

Zwaartekracht: oplossing 3

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

Zwaartekracht: opdracht 4

Zorg dat de bal goed aan de rechterkant stuitert.

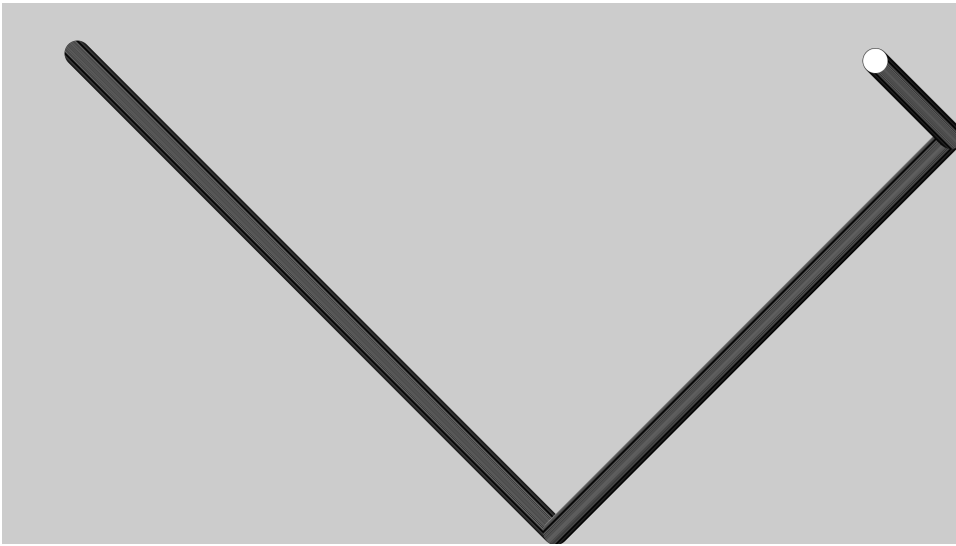


Figure 17: Zwaartekracht: opdracht 4

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

Zwaartekracht: eindopdracht

Voeg onderaan de `draw` functie toe:

```
snelheid_omlaag += 0.1;
```

Maak de bal twee keer zo groot.

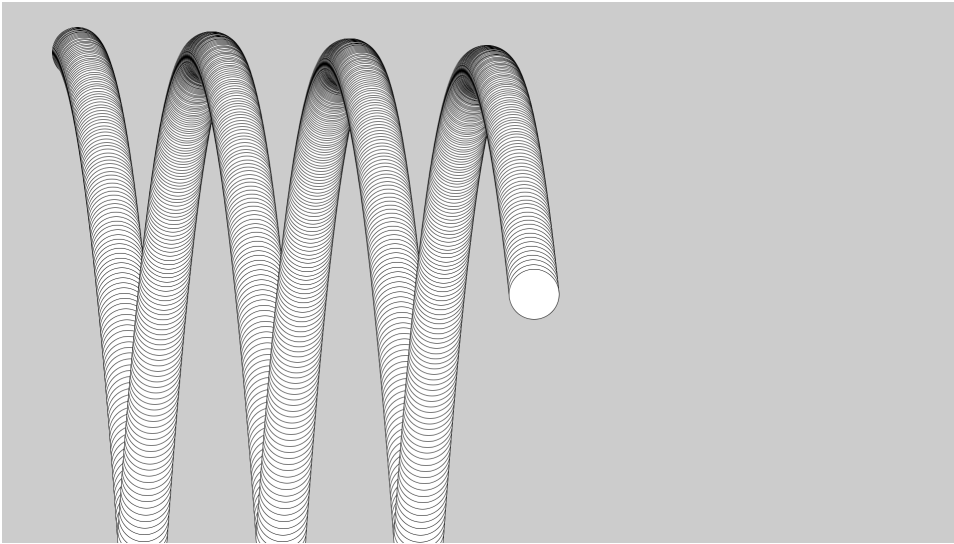


Figure 18: Zwaartekracht: eindopdracht

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



Soms gebeuren er onverwachte dingen in programmeren!

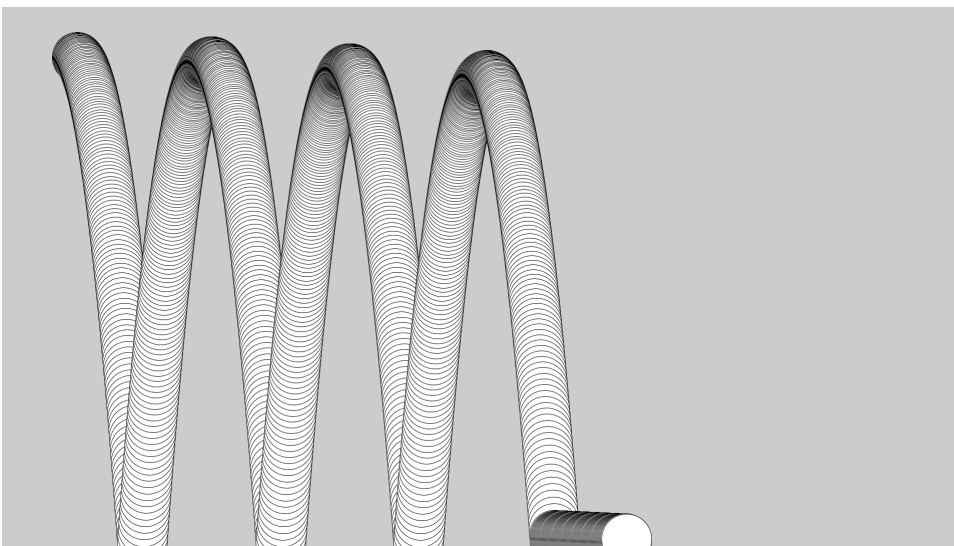


Figure 19: Zwaartekracht eindopdracht ook goed

Arrays 1

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

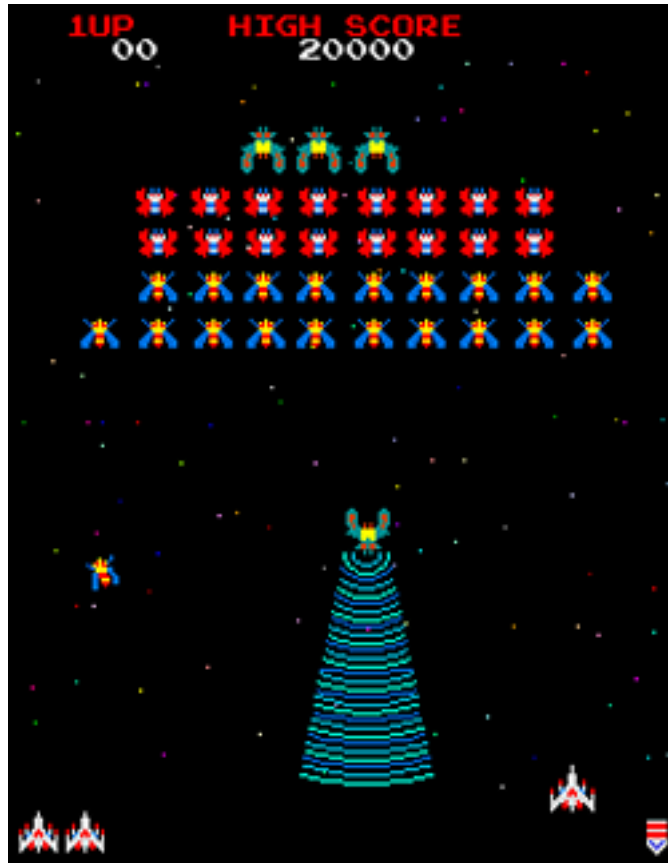


Figure 20: Galaga is een beroemde game met veel vijanden en kogels

Arrays 1: 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;
  }
}
```

Arrays 1: oplossing 1

Een bal die eeuwig naar rechts gaat!

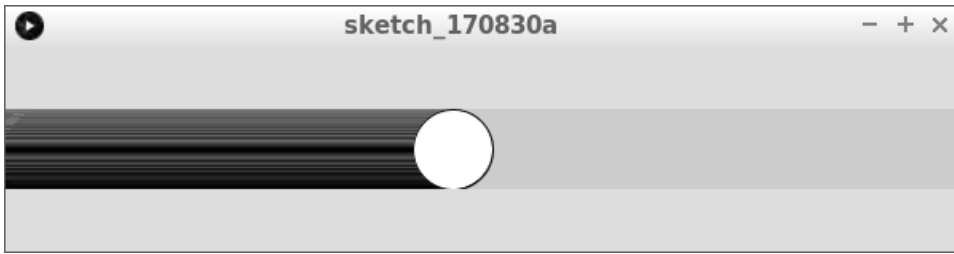


Figure 21: Arrays 1: oplossing 1

Arrays 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`

Arrays 1: 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;
    }
}
```



Dit was zeven regels extra werk

Arrays 1: opdracht 3

Zorg dat er een derde bal bijkomt.

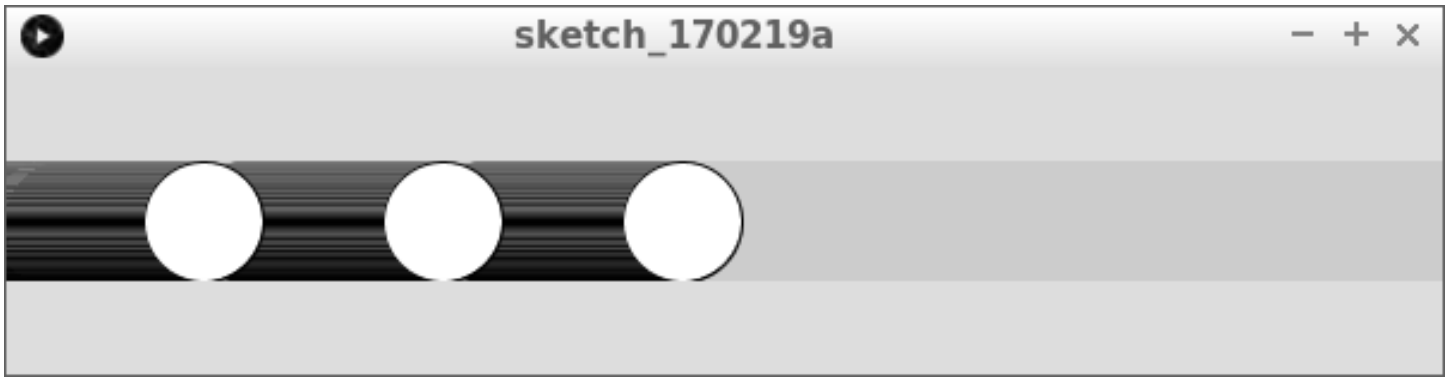


Figure 23: Arrays 1: opdracht 3

Arrays 1: 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!

Arrays 1: wat is een array?

Een array is als een kast met laatjes.



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:

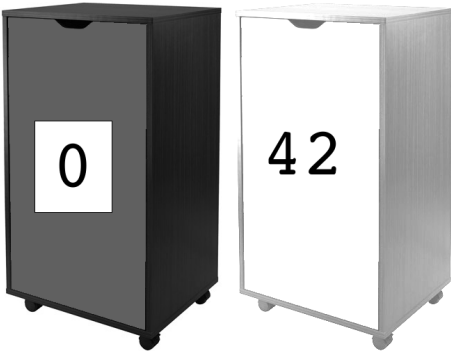


Figure 25: Kast met genummerde laatjes



Het eerste laatje van een array heeft nummer nul

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



plek in array met index nul 'het eerste plekje in de array'

Arrays 1: werken met een array

geheime_getallen

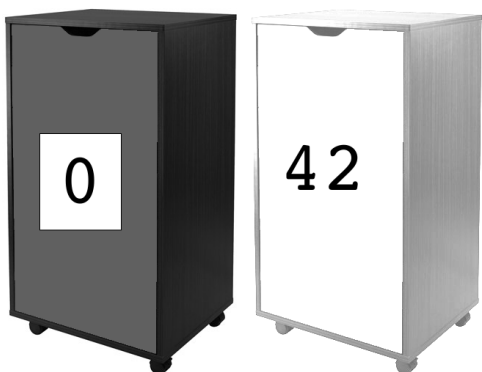




Figure 26: Array met naam 'geheime_getallen' en een laatje

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;
```



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

	
<code>float[] geheime_getallen</code>	'Lieve computer, onthoud keiveel gebroken getallen met de naam <code>geheime_getallen</code> '

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.

	
<code>geheime_getallen = new float[1]</code>	'Lieve computer, maak <code>geheime_getallen</code> 1 laatje groot'

Om de kast met de laatjes precies 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.



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`.



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.

Arrays 1: 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;
    }
}
```


Arrays 1: oplossing 4

Hee, hetzelfde als net!

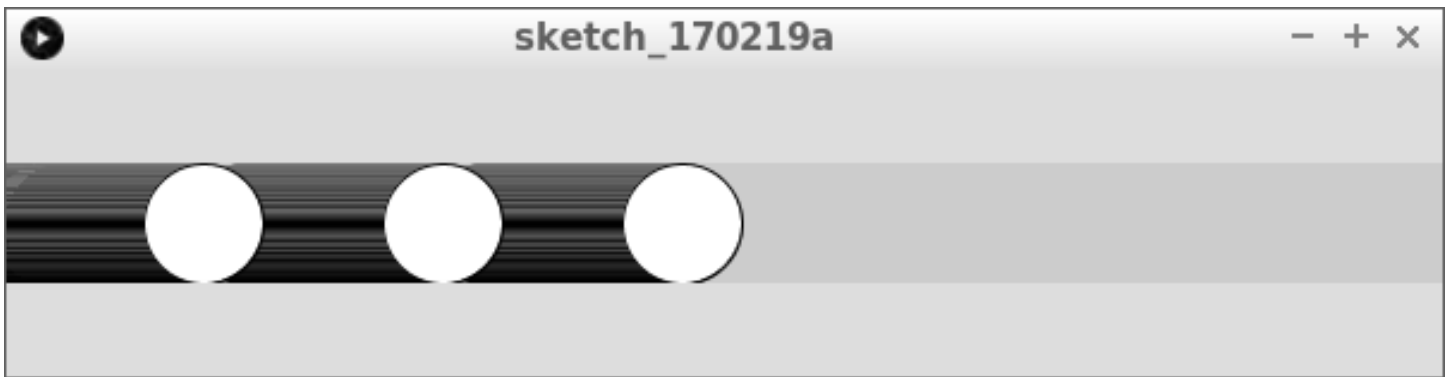


Figure 27: Arrays 1: oplossing 4

Arrays 1: opdracht 5

Run deze code:

```
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;  
        }  
    }  
}
```



Goede programmeur gebruiken liever `for` loops dan dom te knippen
en plakken

Arrays 1: oplossing 5

Hee, hetzelfde als net!



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

Arrays 1: opdracht 6

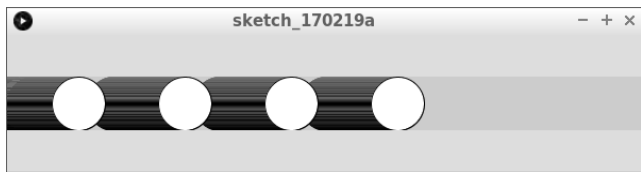


Figure 28: Arrays 1: opdracht 6

Maak nu een vierde bal erbij.



Tip: maak van een 3 een 4

Arrays 1: 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;
        }
    }
}
```

Arrays 1: 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: Arrays 1: opdracht 7

Arrays 1: 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;
    }
  }
}
```

Arrays 1: eindopdracht



Figure 30: Arrays 1: eindopdracht

Maak de code nu zo dat:

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

Arrays2

Nog meer arrays!

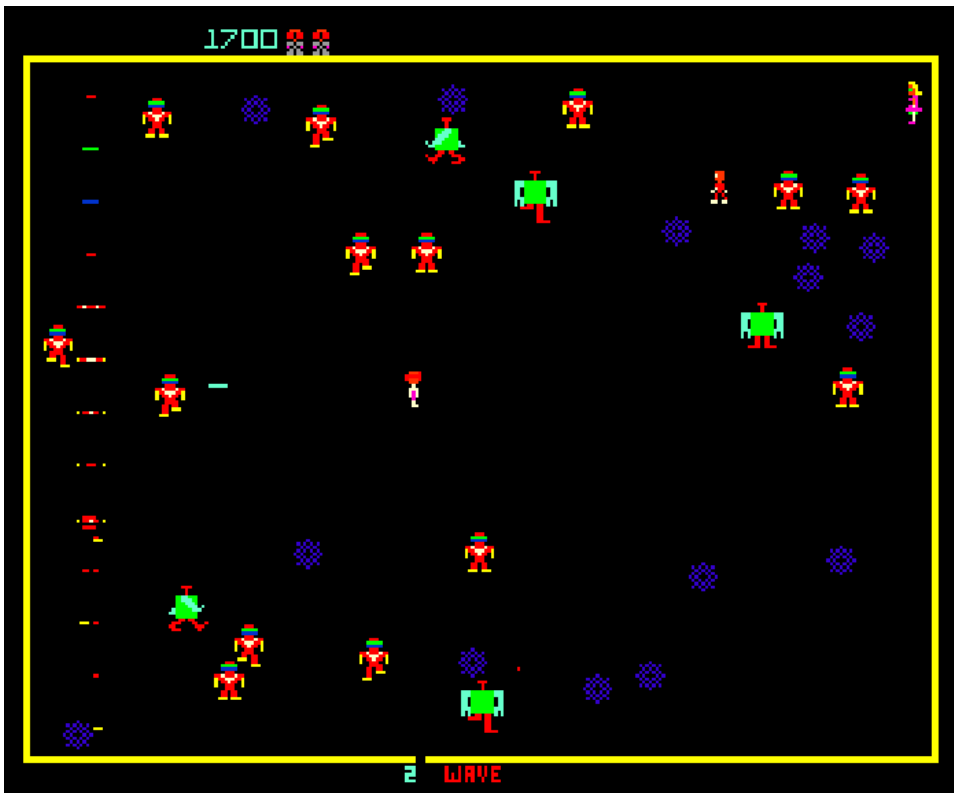


Figure 31: Robotron

Arrays 2: opdracht 1

Run deze code:

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

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

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

Arrays 2: oplossing 1

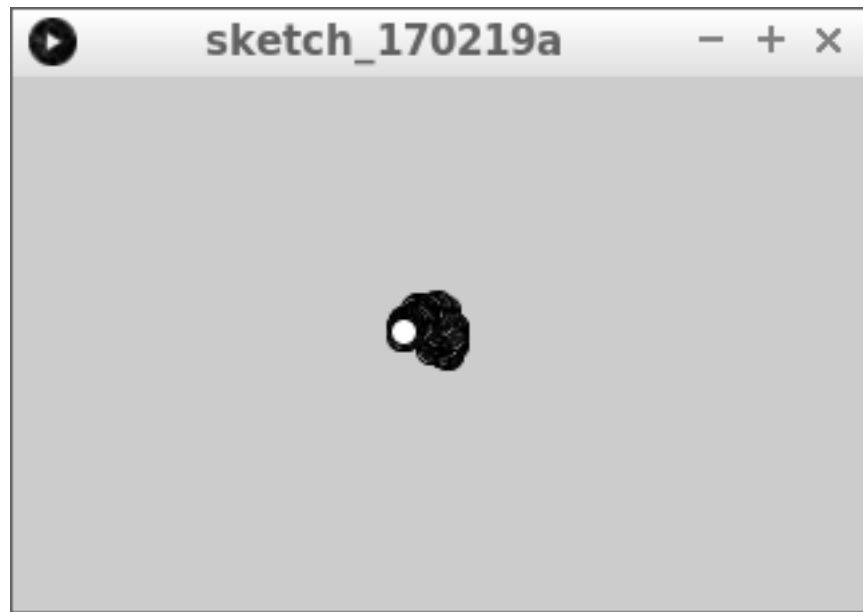


Figure 32: Arrays 2: oplossing 1

Ha, een rookdeeltje.

Arrays 2: opdracht 2

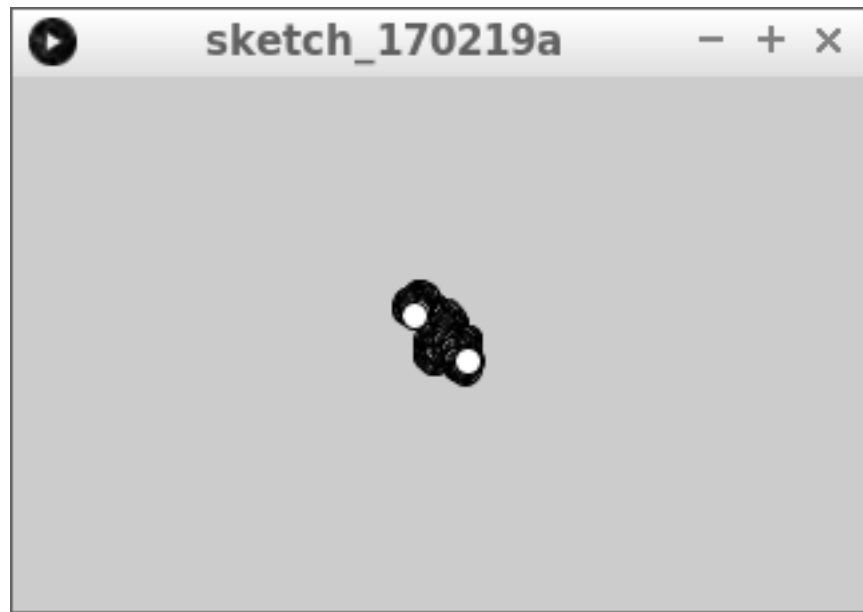


Figure 33: Arrays 2: opdracht 2

Maak een tweede rookdeeltje.

Arrays 2: oplossing 2

```
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);
}
```

Arrays 2: opdracht 3

Gebruik nu een array, zonder for loop.



`float[] xs`

‘Lieve computer, onthoud keiveel gebroken getallen met de naam **xs**’



`xs = new float[2]`

‘Lieve computer, maak **xs** 2 laatjes groot’



Tip: gebruik `xs[0]` inplaats van `x1` en `xs[1]` inplaats van `x2`

Arrays 2: oplossing 3

```
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;
}

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);
}
```

Arrays 2: opdracht 4



Figure 34: Arrays 2: opdracht 4

Gebruik nu for loops. Maak zowel array `xs` en `ys` drie laatjes groot.



```
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’

Arrays 2: oplossing 4

```
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);
    }
}
```

Arrays 2: opdracht 5



Figure 35: Arrays 2: opdracht 5

Elk rookdeeltje krijgt nu een eigen rode randkleur:

- Maak een derde array genaamd **rs**, voor de roodheid van de rookdeeltjes
- In **rs** moeten de getallen 0, 64, 128 en 196 komen
- De roodheid moet ook steeds een meer of minder worden
- De rand van het eerste rookdeeltje, moet de eerste roodheid krijgen. Tip: gebruik **stroke**



Tip: gebruik **stroke** voor de randkleur

Arrays 2: oplossing 5

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

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

Arrays 2: eindopdracht

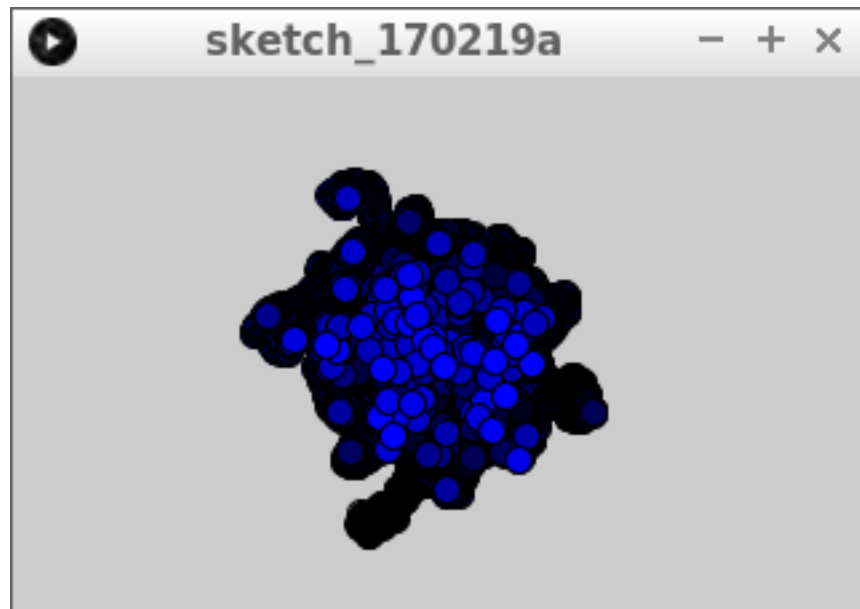


Figure 36: Arrays 2: eindopdracht

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
- niet de rand, maar de vulkleur is blauw



Tip: gebruik `fill` voor de vulkleur
