Figure 1: Structuur

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

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

Figure 3: De licensie van dit boek

(C) Dojo Groningen 2016

Het is nog een beetje een slordig boek. Zo staat bijvoorbeeld het plaatje dat eigenlijk op de kaft moet staan op pagina twee. Er zitten tiepvauten in en de opmaak is ni`et altij`d *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.

Bal naar rechts

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

Het ziet er zo uit:

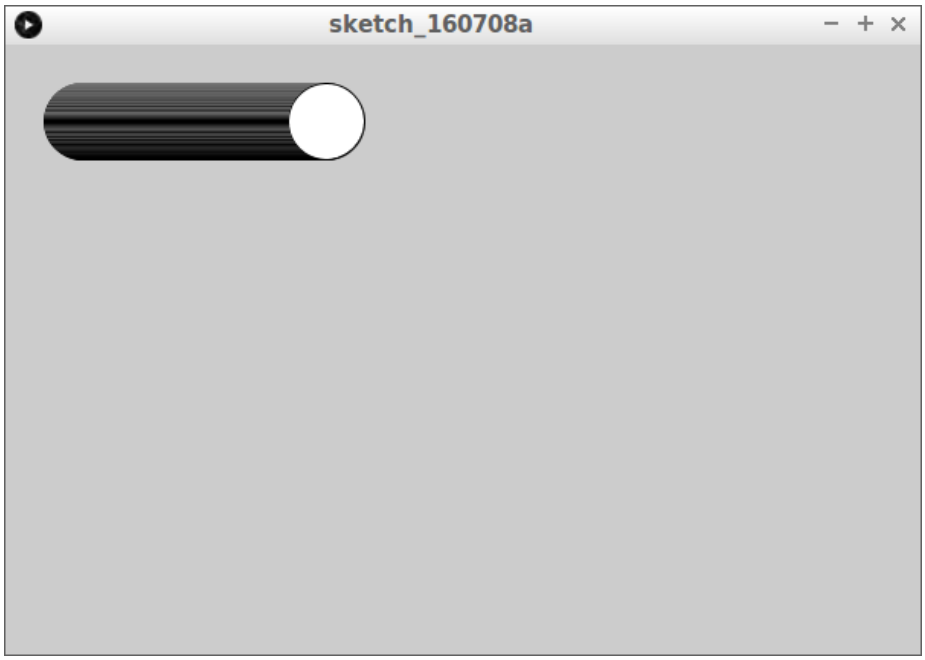


Figure 4: Bal naar rechts

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

Wat weten we al?

Als je de vorige lessen hebt gedaan, weet je wat deze code doet:

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

void draw()
{
  ellipse(50,50,50,50);
}
```

Vragen

- 1. Wat doet dit programma?

- 2. Waar wordt deze cirkel getekend?
- 3. Komt deze cirkel tegen de rand aan?

Oplossing

- 1. Het programma maakt een scherm van 600 pixels breed en 400 pixels hoog. Dan wordt er een cirkel getekent met een middelpunt op coordinaat (50,50) en vijftig pixels breed en hoog
- 2. In de linkerbovenhoek
- 3. Nee, de cirkel komt maar voor de helft van middelpunt tot de rand

Code met een variabele

We gaan de code aanpassen:

```
float x = 50;

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

void draw()
{
  ellipse(x,50,50,50);
}
```

Vragen

- 1. Wat doet dit programma?
- 2. Wat zijn de verschillen?

Oplossing

- 1. Het programma maakt een scherm van 600 pixels breed en 400 pixels hoog. Dan wordt er een cirkel getekent met een middelpunt op coordinaat (50,50) en vijftig pixels breed en hoog
- 2. Er zijn geen verschillen!

Variabelen

De eerste nieuwe regel is:

```
float x = 50;
```

In mensentaal is dit: 'Lieve computer, onthoud het getal `x`. `x` heeft een beginwaarde van vijftig.'.

Een variabele is iets dat onthouden moet worden. Een kassa onthoudt bijvoorbeeld de hoeveelheid geld die alle boodschappen bij elkaar zijn. 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.

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 getal is. Het symbool `=` betekent 'wordt vanaf nu'. Het getal `50` is de beginwaarde.

De tweede veranderde regel is:

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

In mensentaal is dit: 'Lieve computer, teken een ovaal die:

- `x` naar rechts is. De computer weet nog wel wat `x` is: vijftig!
- `50` omlaag is
- `50` pixels breed is
- `50` pixels hoog is

Vragen

- 1. Wat als je `float x = 50;` weghaalt?
- 2. Wat als je `float x = 50;` verandert naar `float rechts = 50;`?
- 3. Wat als je `float x = 50;` verandert naar `float x = 100;`?
- 4. Wat als je `ellipse(x,50,50,50);` weghaalt?
- 5. Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(rechts,50,50,50);`?
- 6. Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,50,50);`?
- 7. Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,50);`?
- 8. Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,x);`?
- 9. Wat als je `x` vervangt door `dinosaurus`?

Oplossing

- 1. Dan komt er onderin 'x cannot be resolved to variable' te staan. Het programma doet het niet meer. Dit omdat de computer `x` niet meer kan vinden
- 2. Dan komt er onderin 'x cannot be resolved to variable' te staan. Het programma doet het niet meer. Dit omdat de computer `x` niet meer kan vinden
- 3. Dan komt het middelpunt van de cirkel honderd pixels naar rechts te liggen
- 4. Dan wordt er geen cirkel meer getekend
- 5. Dan komt onderin 'The variable "rechts" does not exist' te staan. De computer kent geen variabele met de naam `rechts`
- 6. Dan komt het middelpunt van de cirkel honderd pixels naar rechts en honderd pixels naar onder te liggen
- 7. Dan komt het middelpunt van de cirkel honderd pixels naar rechts en honderd pixels naar onder te liggen. De cirkel wordt honderd pixels breed
- 8. Dan komt het middelpunt van de cirkel honderd pixels naar rechts en honderd pixels naar onder te liggen. De cirkel wordt honderd pixels breed en hoog
- 9. Dan komt onderin 'The variable "dinosaurus" does not exist' te staan. De computer kent geen variabele met de naam `dinosaurus`

Bewegen

Nu gaan we de cirkel laten bewegen:

```
float x = 50;

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

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

Vragen

- Wat doet dit programma?
- Wat zijn de verschillen?

De nieuwe regel is:

```
x = x + 1;
```

In mensentaal is dit: 'Lieve computer, x is vanaf nu x plus een'. Of: 'Maak x een hoger'.

Vragen

- Als `x` vijftig is, wat is `x` dan na `x = x + 1;`?
- Als `x` 51 is, wat is `x` dan na `x = x + 1;`?
- Als `x` 52 is, wat is `x` dan na `x = x + 1;`?
- Als `x` 53 is, wat is `x` dan na `x = x + 1;`?
- Als `x` 54 is, wat is `x` dan na `x = x + 1;`?

Nu kunnen we snappen wat het programma doet. Hier staat het programma weer:

```
float x = 50;

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

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

De eerste keer dat de computer `draw` gaat doen, dan vult deze voor `x` een `50` in. Daarna wordt x een hoger. Dan is `draw` klaar.

De tweede keer dat de computer `draw` gaat doen, dan vult deze voor `x` een `51` in. Daarna wordt x een hoger. Dan is `draw` klaar.

De derde keer dat de computer `draw` gaat doen, dan vult deze voor `x` een `52` in. Daarna wordt x een hoger. Dan is `draw` klaar.

Vragen

- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,50,50);`?

- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,50,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,50,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,50,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,50,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,x,50);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(50,x,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,50,x,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,50,x);`?
- Wat als je `ellipse(x,50,50,50);` vervangt door `ellipse(x,x,x,x);`?
- Wat als je `x = x + 1;` vervangt door `x = x + 2;`?
- Wat als je `x = x + 1;` vervangt door `x = x + 10;`?
- Wat als je `x = x + 1;` vervangt door `x = x + 0;`?
- Wat als je `x = x + 1;` vervangt door `x = x - 1;`?
- Wat als je `x = x + 1;` vervangt door `x = x - 0;`?
- Wat als je `x = x + 1;` vervangt door `x = x * 2;`?
- Wat als je `x = x + 1;` vervangt door `x = x * 1;`?
- Wat als je `x = x + 1;` vervangt door `x = x * 0;`?
- Wat als je `x = x + 1;` vervangt door `x = x / 2;`?
- Wat als je `x = x + 1;` vervangt door `x = x / -2;`?
- Wat als je `x = x + 1;` vervangt door `x = x / 1;`?
- Wat als je `x = x + 1;` vervangt door `x = x / 0;`?

Bal naar links

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

Opdracht

- Laat een bal aan de rechterkant van het scherm beginnen
- De bal moet in een rechte lijn naar links gaan

Oplossing

```
float x = 500;

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

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

Bal naar onder

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

Opdracht

- Verander de naam van de variabele `x` in `y`
- Laat een bal aan de bovenkant van het scherm beginnen
- De bal moet in een rechte lijn naar onder gaan

Oplossing

```
float y = 50;

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

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

Bal snel omhoog

Nu gaan we de bal sneller laten bewegen

Opdracht

- 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

```
float y = 50;

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

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

Bal groter

Nu gaan we de bal sneller bewegen en groter maken

Opdracht

- Gebruik als variabele naam `x`
- Laat een bal aan de linker van het scherm beginnen
- De bal moet in een rechte lijn naar rechts gaan
- De bal moet even groot blijven

Oplossing

```
float x = 50;

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

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

```
}
```

Bal veranderd van kleur

Nu gaan we de bal een kleur geven met `fill`

Opdracht

- Gebruik als variabele naam `t` (van tijd) in plaats van `x` of `y`
- Laat een bal aan de linker van het scherm beginnen
- De bal moet in een rechte lijn naar rechts gaan
- De kleur moet van zwart naar wit veranderen

Oplossing

```
float t = 50;

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

void draw()
{
  fill(t,t,t);
  ellipse(50,t,50,t);
  t = t + 1;
}
```

Eindopdracht

- gebruik een variabele `t` (van tijd)
- de bal moet snel omlaag en naar rechts gaan
- de bal moet groter worden in de breedte en hoogte
- de kleur moet van zwart naar wit veranderen

Verder

Je zou nu kunnen doen:

- Bal die eeuwig naar rechts gaat

Bal die eeuwig naar rechts gaat

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

Het ziet er zo uit:

Figure 5: Bal die eeuwig naar rechts gaat 1

Figure 6: Bal die eeuwig naar rechts gaat 2

We leren in deze les wat `if`-statement is. Je kunt (bijna) niet programmeren zonder `if`-statements.

Wat weten we al?

Als je de vorige lessen hebt gedaan, weet je wat deze code doet:

```
float x = 50;

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

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

Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal
- Blijft de ovaal zichtbaar op het scherm?
- Kopieer de code en bekijk het programma. Klopt wat je dacht?

Een `if`-statement

We willen kunnen zeggen: 'Lieve computer, *als* de bal te ver naar rechts is, dan teleporteer je de bal naar rechts'. `if` is Engels voor 'als'.

Zo zou dit kunnen:

```
if (x > 200)
{
  x = 100;
}
```

Dit betekent:

- `if`: begin van een if statement. Een if-statement heeft dan twee gedeeltes:
- `()`: tussen de ronde haken staat een test; iets wat waar of niet waar is
- `{}`: tussen de accolades staat wat de computer moet doen als de test waar is
- `x > 200`: dit staat tussen de ronde haken. Dit is de test 'x is groter dan 200'. Het `>` tekentje betekent 'groter dan'
- `x = 100`: dit staat tussen de accolades. Als 'x is groter dan 200' waar is, dan krijgt `x` de waarde 100

Preciezer zeg je: 'Lieve computer, *als* x meer is dan 200, zet x dat op 100'. `if` is Engels voor 'als'.

Vragen

- Kopieer het `if`-statement tussen de accolades van de `draw` functie
- Wat doet het programma?

Als het kopieren niet is gelukt, gebruik dan deze code:

```
float x = 50;

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

```
void draw()
{
  ellipse(x,100,100,100);
  x = x + 1;
  if (x > 200)
  {
    x = 100;
  }
}
```

- Kun je ervoor zorgen dat de ovaal helemaal naar de linkerkant van het scherm springt?
- Kun je ervoor zorgen dat de ovaal helemaal naar rechst beweegt, voordat deze naar de linkerkant van het scherm springt?

Antwoord

Dit is een eeuwig naar rechts gaande bal:

```
float x = -50;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + 1;
  if (x > 650)
  {
    x = -50;
  }
}
```

Het lijkt al een beetje op Lonelier Pong. Dit is geen toeval :-)

Bal die eeuwig naar links gaat

Gefopt! Ook al het de les 'Bal die eeuwig naar rechts gaat', we gaan toch de bal ook andere kanten op laten gaan.

We gaan nu een bal programmeren die eeuwig naar links gaat.

Wat je nu moet weten is het `if` statement om te zeggen wanneer de `x` te klein is:

```
if (x < 100)
{
  x = 500;
}
```

Hiermee zeg je: 'Lieve computer, als `x` kleiner (`<`, hier kun je een `k` van maken) is dan honderd, zet dan `x` op vijfhonderd.

Opdracht

Maak een bal die eeuwig naar links gaat:

- De bal begint buiten het beeld
- De bal gaat helemaal het beeld uit
- Als de bal net uit het beeld uit, komt 'ie meteen de andere kant weer binnen

Antwoord

Dit is een eeuwig naar links gaande bal:

```
float x = 650;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x - 1;
  if (x < -50)
  {
    x = 650;
  }
}
```

Bal die eeuwig omlaag gaat

We hebben een bal naar rechts en naar links laten bewegen door de `x` coordinaat te veranderen. De bal kan ook naar omlaag en omhoog gaan door de `y` coordinaat te veranderen.

Opdracht

Schrijf een programma waarin een bal eeuwig omlaag gaat:

- maak het scherm 300 pixels breed en 200 pixels hoog
- gebruik een variable met naam `y`
- vervang de code `ellipse(x,50,100,00)` door `elllipse(50,y,100,100)`
- als de bal omlaag uit het scherm gaat, moet de bal weer bovenin komen

Oplossing

```
float y = -50;

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

void draw()
{
  ellipse(50,y,100,100);
  y = y + 1;
  if (y > 250)
  {
    y = -50;
  }
}
```

Bal die schuin gaat

Hoppa, nu we een variabele `x` of een `y` hebben gemaakt, gaan we beiden doen!

Als we code samenvoegen, gelden de volgende regels:

- alles wat boven de `setup` functie staat, moet daar blijven
- alles wat binnen de `setup` functie staat, moet binnen de `setup` functie blijven
- alles wat binnen de `draw` functie staat, moet binnen de `draw` functie blijven

Opdracht

- Voeg de code van 'Bal die eeuwig naar rechts gaat' samen met 'Bal die eeuwig omlaag gaat'
- Verander de code zo dat de bal schuin gaat

Oplossing

```
float x = -50;
float y = -50;

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

void draw()
{
  ellipse(x,y,100,100);
  x = x + 1;
  y = y + 1;
  if (x > 350)
  {
    x = -50;
  }
  if (y > 250)
  {
    y = -50;
  }
}
```

Eindopdracht

Laat de bal nu eeuwig schuin naar linksonder gaan.

Verder

Je zou nu kunnen doen:

- Bal die horizontaal stuitert

Bal die horizontaal stuitert

In deze les gaan we een bal horizontaal laten stuiteren.

Het ziet er zo uit:

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

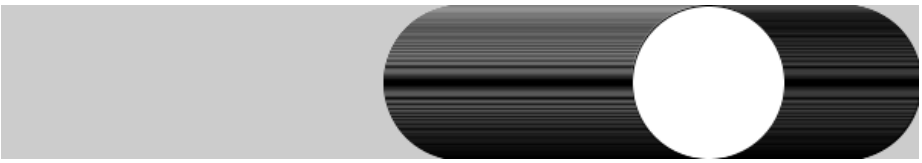


Figure 7: Bal die horizontaal stuitert (zie 'dojo/Images/BalDieHorizontaalStuitertGif')

Wat weten we al?

Dit is een eeuwig naar rechts gaande bal:

```
float x = 300;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + 1;
  if (x > 650)
  {
    x = -50;
  }
}
```

Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal?
- Blijft de ovaal zichtbaar op het scherm?
- Kopieer de code en bekijk het programma. Klopt wat je dacht?
- Verander het programma: laat de bal nu eeuwig naar links gaan

Twee variabelen

Nu onthoudt de computer een variabele: de x-coordinaat van de ovaal. Om een bal te laten stuiteren, moet er ook een richting onthouden worden: de bal gaat immers of naar links of naar rechts.

In deze code wordt de richting van de bal `dx` genoemd. Dit is een afkorting van 'delta x' en dat is weer wiskundetaal voor 'de verandering van x'.

Vragen

```
float x = 300;
float dx = 2;

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    x = -50;
  }
}
```

- In welke richting beweegt de bal?
- Hoeveel pixels beweegt de bal per keer?
- Zet de waarde van `dx` op `1`. Wat zie je?
- Zet de waarde van `dx` op `0`. Wat zie je?
- Zet de waarde van `dx` op `-1`. Wat zie je?
- Bij sommige waarden van `dx` gaat de bal links het beeld uit. Maak een tweede if-statement, die ervoor zorgt dat de bal eeuwig links kan gaan. Test dit bij een `dx` van `2`, `0` en `-2`.
- Wat moet er met de `dx` gebeuren om de bal te laten stuiteren? Probeer dit!

Stuiteren

Als een bal met een snelheid van drie pixels naar rechst gaat en stuitert, dan gaat deze vanaf dan drie pixels naar links. Andersom is dat ook zo.

Hier is een manier om de bal te laten stuiteren:

```
float x = 300;
float dx = 2;

void setup()
```

```
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    dx = -2;
  }
  if (x < 50)
  {
    dx = 2;
  }
}
```

Vragen

- Kopieer en run de code. Stuitert de bal?
- Verander de snelheid van de bal naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal naar drie pixels per keer. Op hoeveel plekken moet je de code aanpassen?
- Verander de snelheid van de bal terug naar een pixel per keer. Op hoeveel plekken moet je de code aanpassen? Laat nu de bal in het begin naar links gaan. Op hoeveel plekken moet je de code aanpassen?

De richting omklappen

Er is een slimmere manier om `dx` te veranderen. We hebben gezien dat als `dx` gelijk was aan `2`, deze `-2` wordt bij bij een stuiter. We hebben gezien dat als `dx` gelijk was aan `-2`, deze `2` wordt bij bij een stuiter. Er komt een minnetje voor, of er komt een minnetje bij. Dit is gemakkelijk op dezelfde manier te doen:

```
dx = -dx;
```

Hiermee zeg je 'Lieve computer, de nieuwe waarde van `dx` is min de oude waarde'. Als de oude waarde van `dx` `2` is, dan wordt deze nu `-2`. Als de oude waarde van `dx` `-2` is, dan wordt deze nu `--2` (jawel, min min twee) en dat mag je schrijven als `2` (omdat min keer min is plus).

Eindopdracht

- Gebruik nu de slimme manier om een bal te laten stuiteren.

Het lijkt al een beetje op Lonelier Pong. Dit is geen toeval :-)

Zwaartekracht

In deze les gaan we zwaartekracht programmeren.

Het ziet er zo uit:

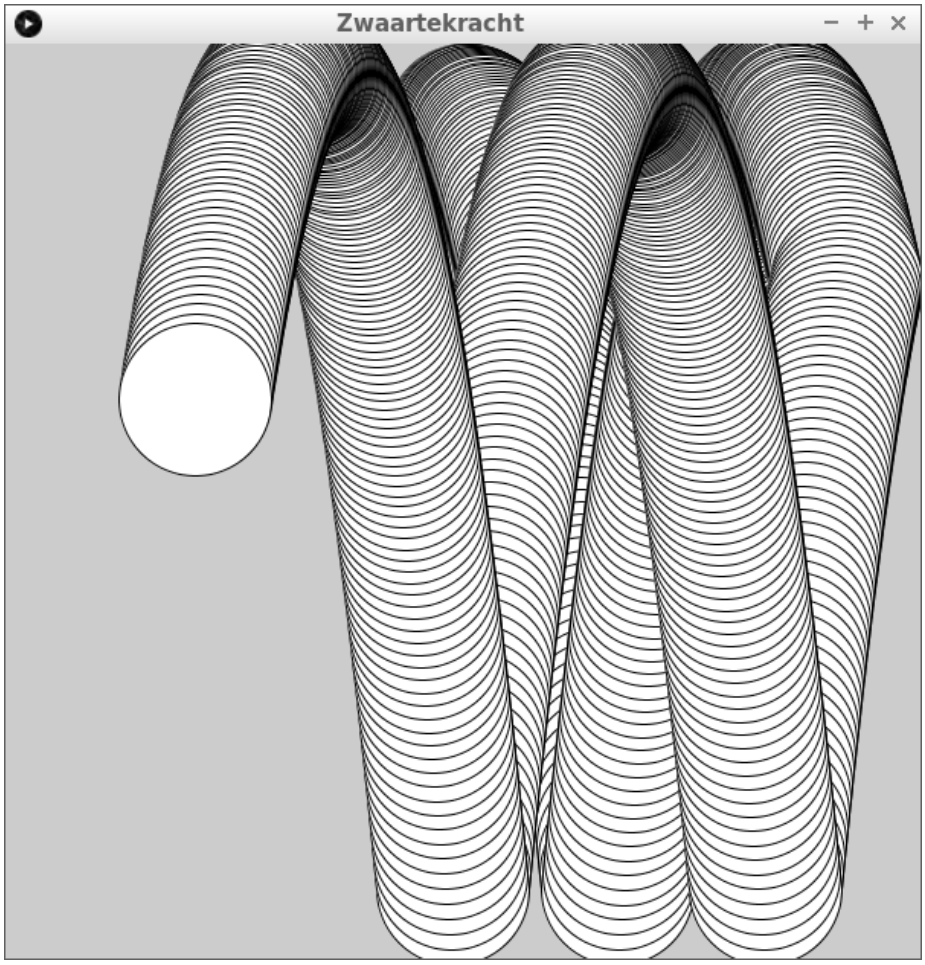


Figure 8: Zwaartekracht

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

Wat weten we al?

Dit is een eeuwig horizontaal stuiterende bal:

```
float x = 300;
float dx = 1; //Snelheid in de x richting

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 600 || x < 0)
  {
    dx = -dx;
  }
}
```

Vragen

- 1. Wat doet dit programma?
- 2. In welke richting beweegt de ovaal in het begin? In welke regel zie je dat?
- 3. Blijft de ovaal zichtbaar op het scherm?
- 4. Zorg ervoor dat de ovaal netjes op het scherm zichtbaar blijft. Hij moet 100 bij 100 pixels groot blijven

Oplossing

- 1. De bal stuitert horizontaal: eerste gaat 'ie rechts, bij de rand verandert deze van richting naar links. Weer aan de linkerkant, stuitert de bal weer naar rechts
- 2. Naar rechts. Dit zie je aan `x = x + dx` (dit zorgt ervoor dat de bal beweegt) en `float dx = 1`. Hierdoor wordt de eerste regel dus `x = x + 1`. Als `x` meer wordt, gaat de bal meer naar rechts

- 3. Nee, de ovaal gaat een stuk het veld uit

```
float x = 300;
float dx = 1; //Snelheid in de x richting

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

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 550 || x < 50)
  {
    dx = -dx;
  }
}
```

Gas geven

Nu gaat een bal altijd `dx` naar links of naar rechts. Een bal kan ook steeds sneller gaan. Dit kun je doen door `dx` te veranderen!

Zet nu deze code in Processing:

```
float x = 50;
float dx = 0; //Snelheid in de x richting
float a = 1; //Versnelling

void setup()
{
  size(600, 100);
}

void draw()
{
  ellipse(x,50,100,100);
  x = x + dx;
  dx = dx + a;
  if (x > 650)
  {
    x = 50;
    dx = 0;
  }
```

```
}
```

Vragen

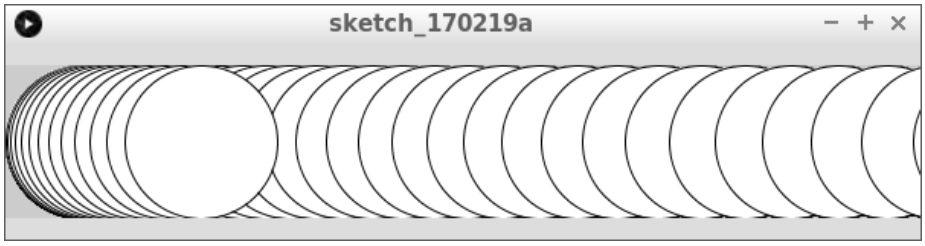


Figure 9: Gas geven

- 1. Wat zie je gebeuren?
- 2. Waarom staat er `float dx = 0`?
- 3. In het `if` statement staat `x = 50`. Wat doet dit?
- 4. In het `if` statement staat `dx = 0`. Wat doet dit?
- 5. Verander de code in het `if` statement van `dx = 0` naar `dx = -dx`. Wat doet dit?

Oplossing

- 1. De bal gaat naar rechts. De bal gaat ook steeds sneller naar rechts. Als de bal rechts is, begint deze weer links
- 2. Dan staat de bal in het begin stil
- 3. Deze zet de bal weer naar links
- 4. Deze zet de bal weer stil
- 5. De bal gaat nu stuiteren als een stuiterbal

Kleur veranderen

Nu gaan we de bal iets extra's laten doen: van zwart naar rood laten verkleuren.

Opdracht

- Laat de bal nog steeds stuiteren tegen de rechterkant van het scherm

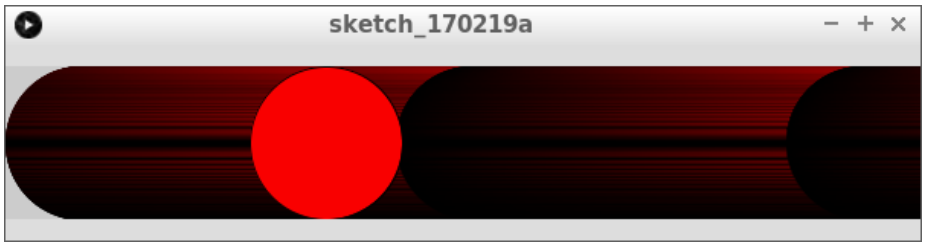


Figure 10: Opdracht 'Kleur veranderen'

- Maak een extra variabele aan met naam `r` (van `rood`)
- `r` begint op nul en wordt steeds een hoger
- Als `r` groter is dan 255, dan wordt `r` nul
- `r` bepaalt de kleur van de bal:
  - als `r` nul is, is de bal zwart
  - als `r` 255 is, is de bal rood

Oplossing

```
float x = 50;
float dx = 1; //Snelheid in de x richting
float r = 0; //Roodheid

void setup()
{
  size(600, 100);
}

void draw()
{
  fill(r, 0, 0);
  ellipse(x,50,100,100);
  x = x + dx;
  if (x > 650)
  {
    x = 50;
  }
  r = r + 1;
  if (r > 255)
  {
    r = 0;
  }
}
```

Zwaartekracht

De zwaartekracht trekt aan voorwerpen. Iets dat omlaag valt, gaat hierdoor steeds sneller vallen. Iets dat omhoog gaat, gaat eerst steeds langzamer omhoog, en gaat daarna ook vallen. In de natuukunde gebruiken ze `g` (van 'gravity', dit is Engels voor zwaartekracht) voor de zwaartekracht.

Opdracht

- Laat de bal nu als een stuiterbal vallen en stuiteren
- Gebruik de variabele naam `g` inplaats van `a`
- De bal moet verticaal (van omhoog naar omlaag) versnellen
- De bal moet horizontaal (van links naar rechts) even snel blijven gaan

`width` en `height`

`width` en `height` zijn ingebouwde variabelen in Processing, die handig zijn om te gebruiken zodat je programma nog werkt als je de grootte van je scherm aanpast.

Stel dat je een programma maakt wat een ovaal tekent die het scherm opvult, deze zou er zo uit kunnen zien:

```
void setup() {
  size(256, 256);
  ellipse(128, 128, 256, 256);
}
```

Dit programma tekent dit:

Maar dit programma werkt alleen voor een scherm wat 256 bij 256 pixels is. Dat is natuurlijk onhandig, want elke keer als je een nieuwe grootte kiest moet je 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 cordinaat van de ellipse is namelijk de helft van de breedte, de y cordinaat de helft van de hoogte. En de breedte en hoogte van de `ellipse` zijn hetzelfde als die van het scherm.

Gelukkig weet Processing de breedte en hoogte van het scherm. De breedte van het scherm heet in Processing `width` en de hoogte heet `height`. Deze getallen worden bepaald zodra je size gebruikt om de grootte van je scherm te defineren.

Het programma wat een ovaal tekent die het scherm opvult, ziet er dan zo uit:

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

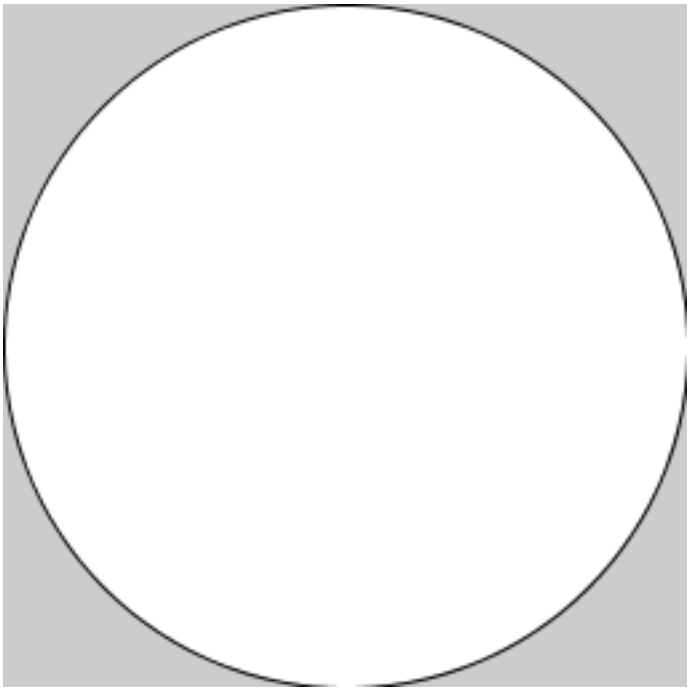
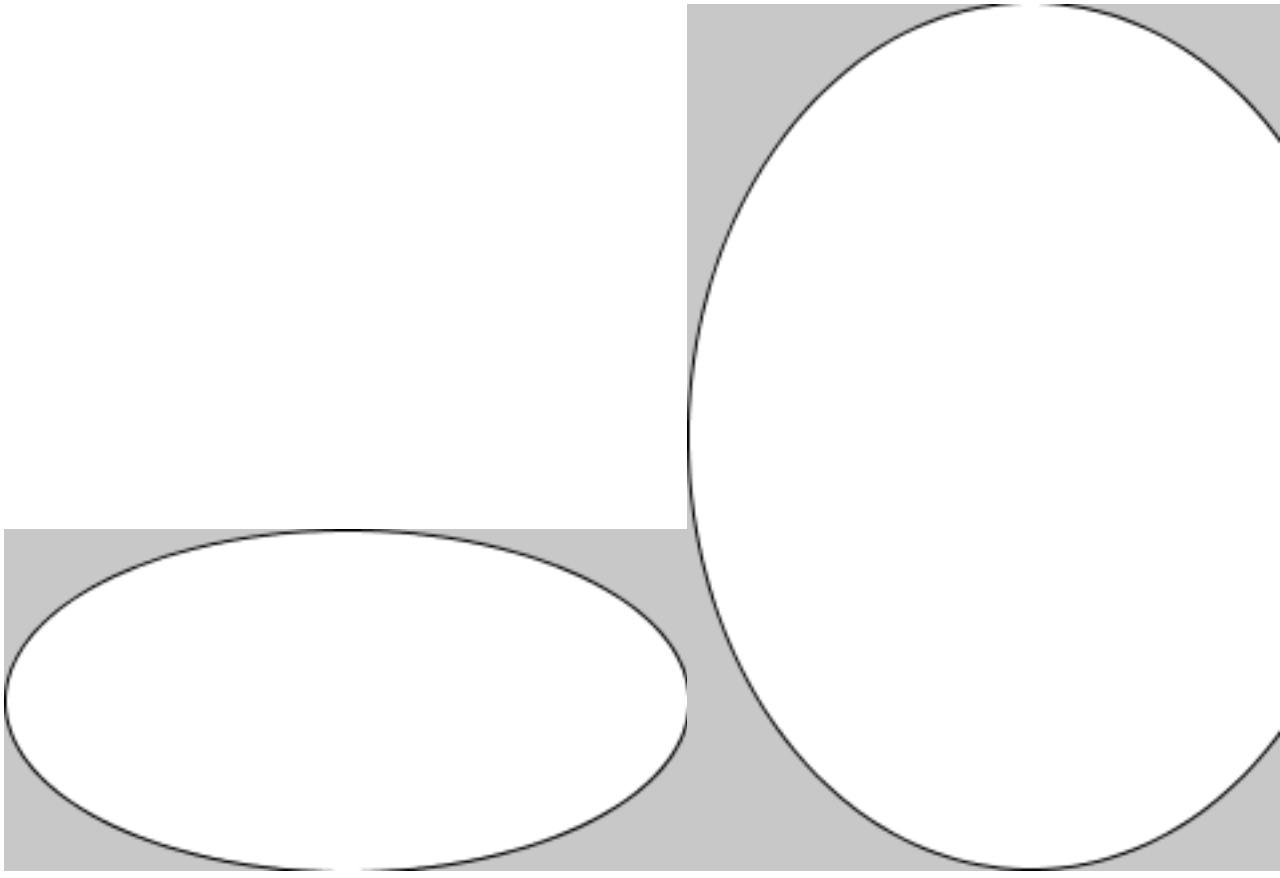


Figure 11: circle 256

```
  ellipse(width/2, height/2, width, height);
}
```

Maar nu past de ovaal nog steeds als we de getallin in `size` veranderen!



EINDOPDRACHT

Teken vier ellipsen met breedte `width` en lengte `heigth`, in de vier hoeken in het scherm.

Arrays1

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:

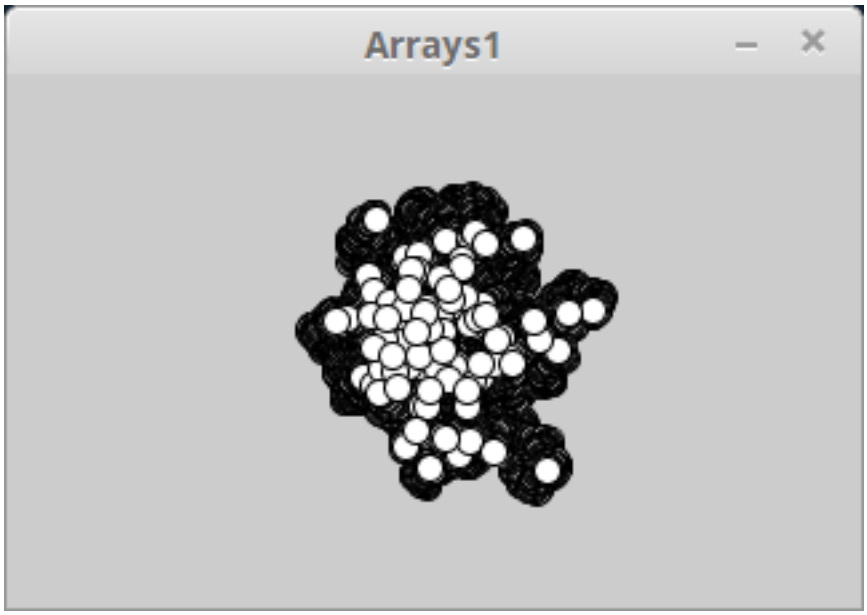


Figure 12: Honderd rookdeeltjes

Twee ballen die eeuwig naar rechts gaan

Je hebt een bal euuwig 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

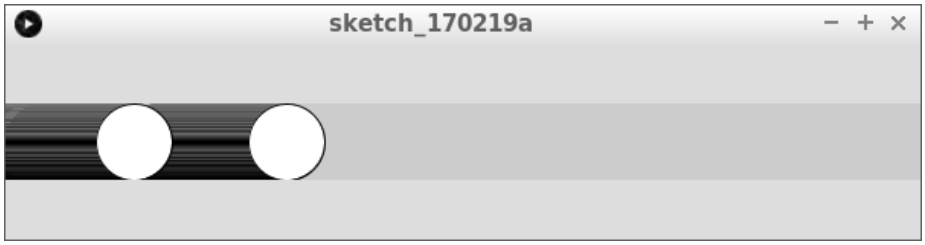


Figure 13: Twee ballen die eeuwig naar rechts gaan

- 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

- 1. Zo doe je dat:

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

void setup()
{
```

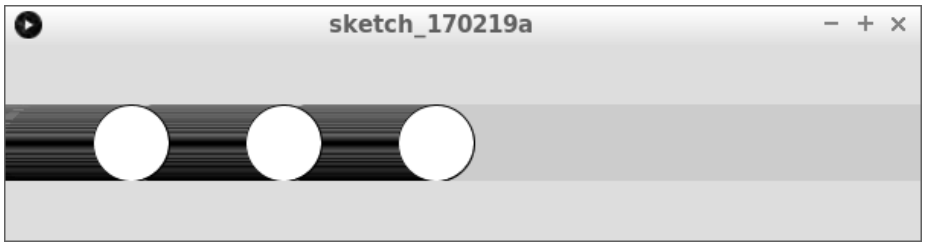


Figure 14: Drie ballen

```
  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:

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 tweeenveertig.

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

Werken met een array met 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;
```

Figure 15: Kast met laatje

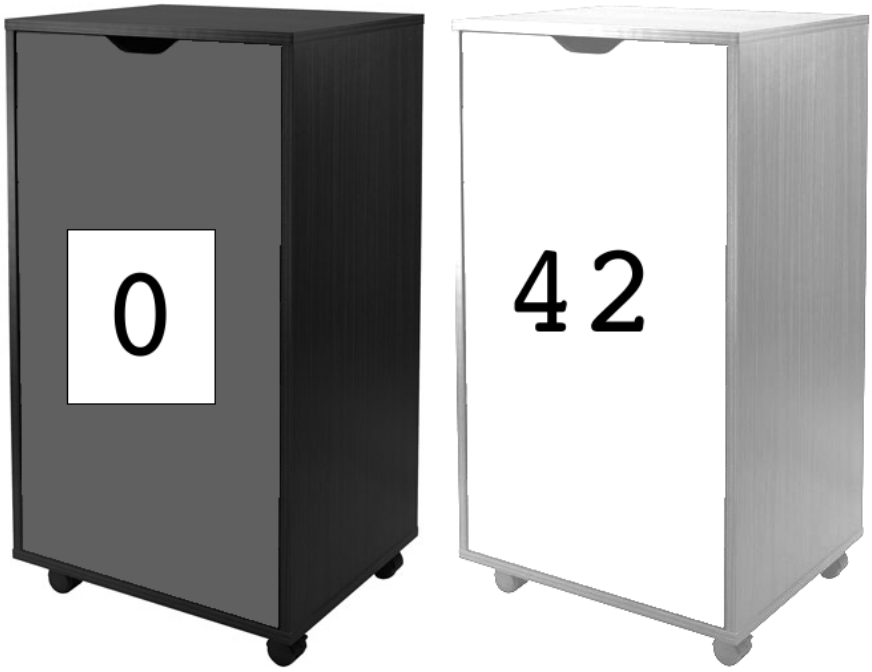


Figure 16: Kast met genummerde laatjes

Hiermee zeg je, in de derde regel: 'Lieve computer, stop in laatje met index nul het getal tweeenveertig'. 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;
}
```

Figure 17: Array met naam ‘geheime_getallen’ en een laatje

```
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:

```
float[] xs;

void setup()
{
```

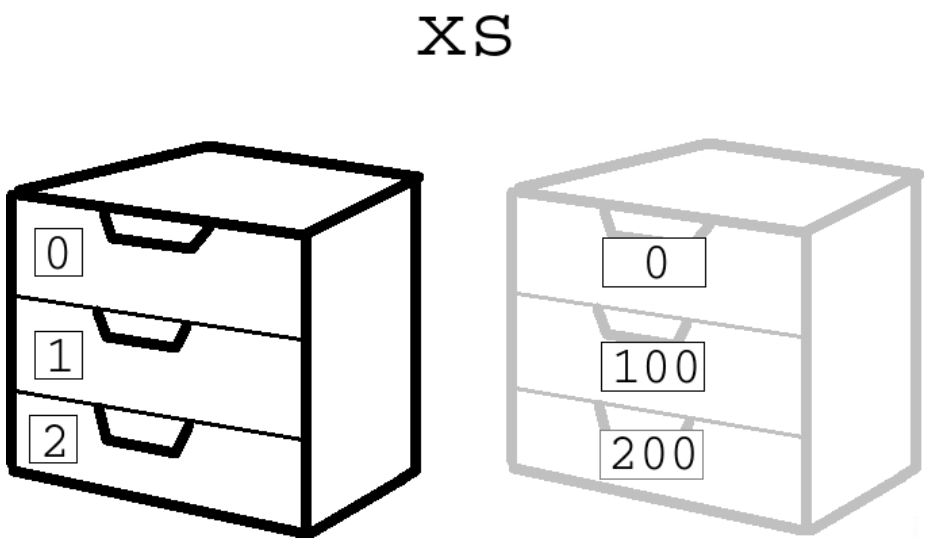


Figure 18: Array voor ‘Drie ballen met een array’

```
  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`

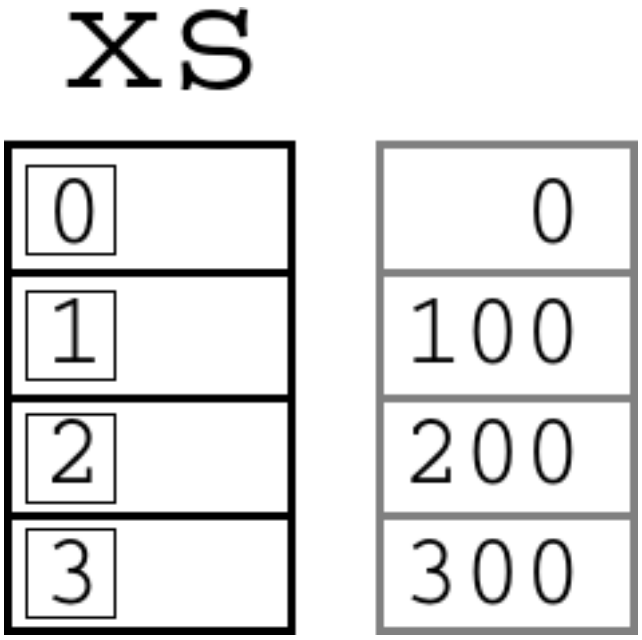


Figure 19: Array met vier laatjes

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:

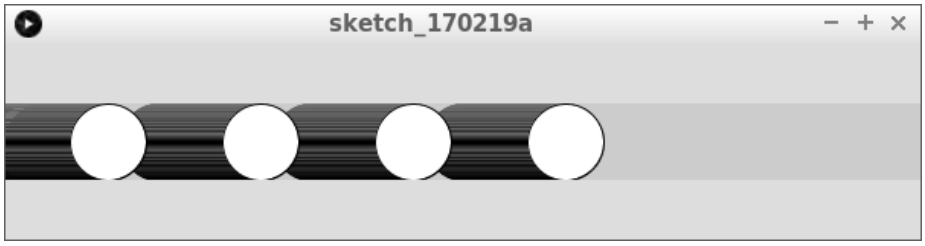


Figure 20: Vier ballen

```
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;
    }
  }
}
```

- 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 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:

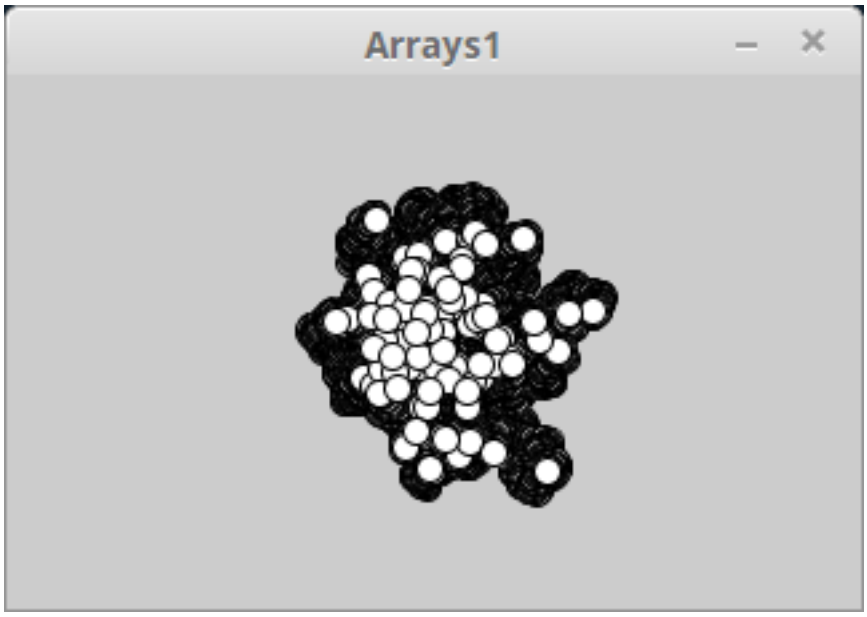


Figure 21: Honderd rookdeeltjes

Een rookdeeltje

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

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

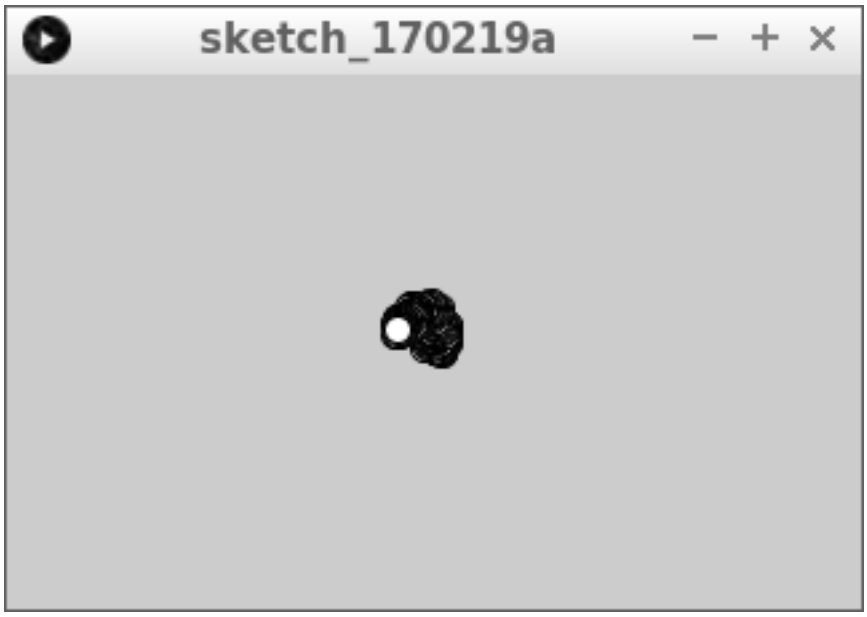


Figure 22: Een rookdeeltje

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:

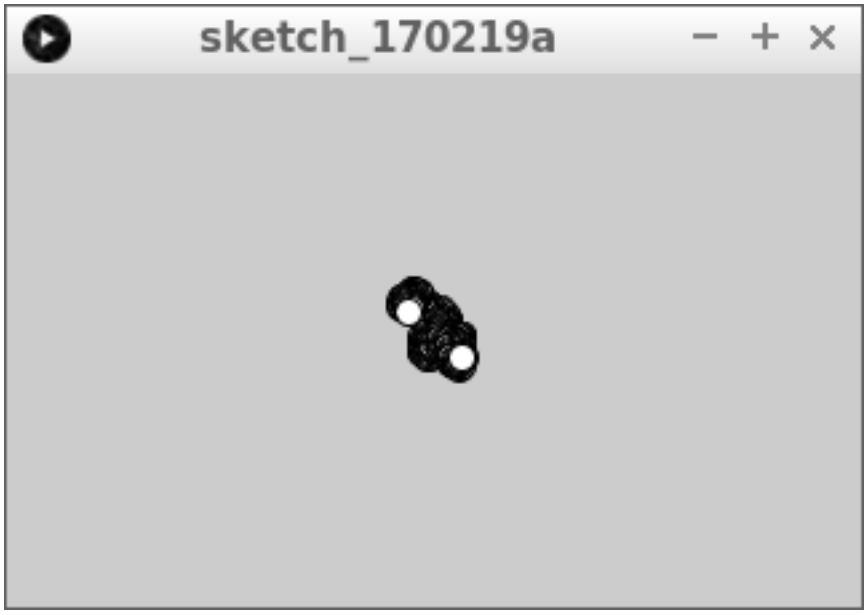


Figure 23: Twee rookdeeltjes

```
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);
}
```

- 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;
}

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);
```

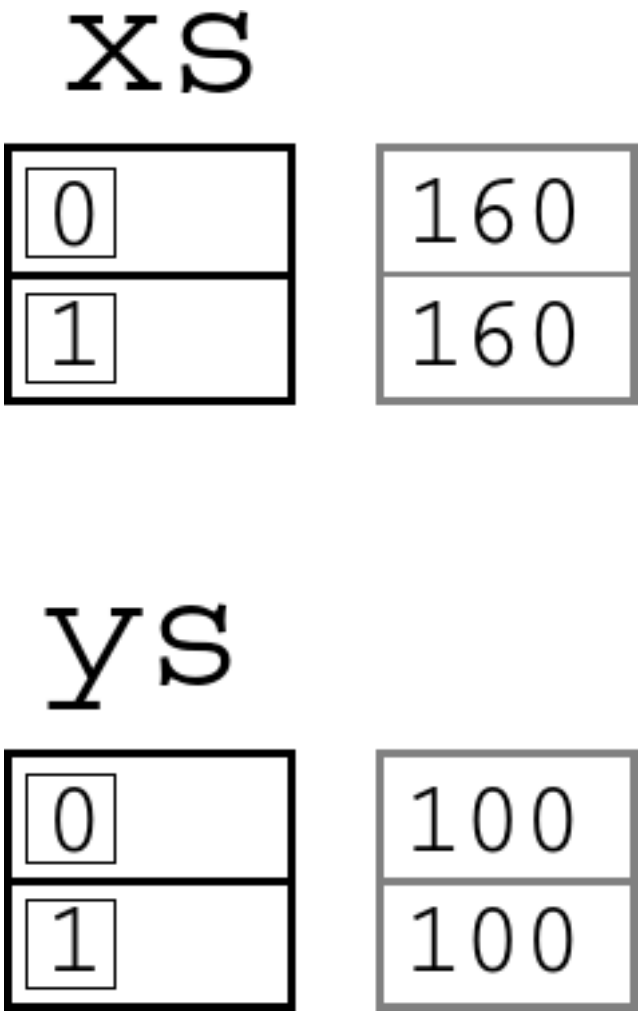


Figure 24: Array van ‘Twee rookdeeltjes met een array, zonder for loop’

```
  ellipse(xs[1], ys[1], 10, 10);
}
```

Drie rookdeeltjes

Nu laten we de code een for loop gebruiken.

Opdracht

Figure 25: 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

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 26: 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);
  }
```

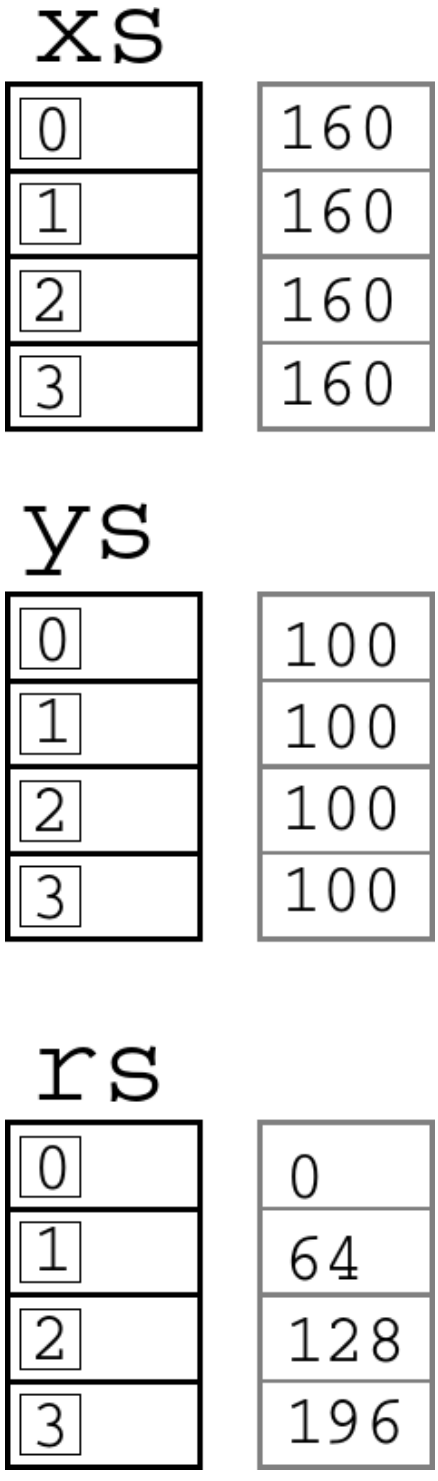


Figure 27: Arrays bij opdracht 'Vier rookdeeltjes'

```
}
```

Eindopdracht

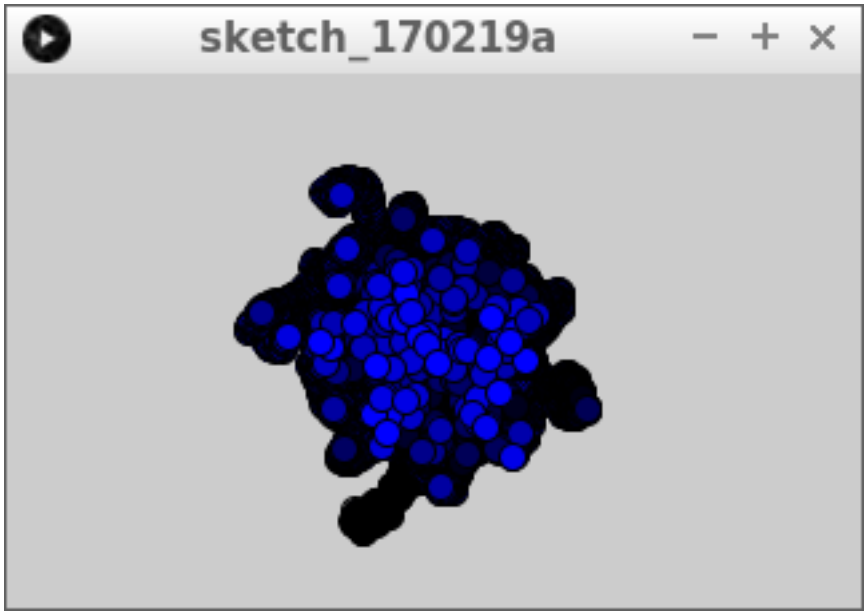


Figure 28: Arrays2 eindopdracht

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`)

PImage les 1

In deze les gaan we met een plaatje werken.

Een plaatje vinden

Op de GitHub van de cursus hebben we een aantal goede plaatjes verzameld.

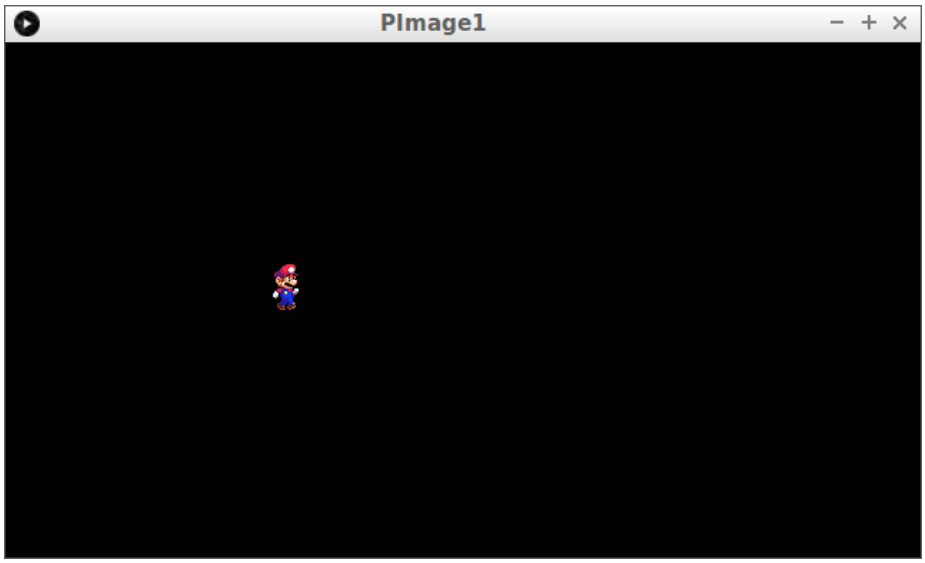


Figure 29: PImage1.png

Er zijn twee manieren de plaatjes te downloaden:

- Ga naar https://github.com/richelbilderbeek/Dojo/tree/master/Sprites. Klik dan op de naam van een plaatje. Nu kun je het plaatje zien. Klik met de rechtermuisknop op het plaatje en kies 'Afbeelding opslaan als'/'Save image as'
- Je kunt alle plaatjes in een keer downloaden. Download de cursus als zip hier: https://github.com/richelbilderbeek/Dojo/archive/master.zip. Kies 'Alles uitpakken'/'Extract all' om het zipje uit te pakken. In de folder 'Sprites' staan de plaatjes

In dit voorbeeld gebruik ik dit plaatje van Mario:

Figure 30: mario.png

Code

De code om Mario op de muis cursor te laten zien is simpel:

```
PImage plaatje;
```

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

void draw() {
  background(0);
  image(plaatje, mouseX, mouseY);
}
```

- `PImage plaatje`: onthoud een `PImage` met de naam `plaatje`. Let op: `PImage` begint met twee hoofdletters
- `void setup() {}`: de setup functie, de computer voert een keer alles tussen de accolades uit
- `size(640, 360)`: maak een scherm van 640 pixels breed en 360 pixels hoog
- `plaatje = loadImage("mario.png")`. Laat `plaatje` de afbeelding van Mario krijgen, door het bestand `mario.png` te laden
- `void draw() {}`: de draw functie, de computer voert steeds alles tussen de accolades uit
- `background(0)`: maak de achtergrond zwart
- `image(plaatje, mouseX, mouseY)`: teken (de linkerbovenhoek van) het plaatje `plaatje` op de plek waar de muiscursus is.

Dit werkt niet meteen, omdat de bestanden op de juiste plek moeten staan.

Bestanden op de juiste plek

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

- 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`

Opdracht

- Krijg bovenstaand voorbeeld werkend
- Maak een eigen programma met een ander plaatje

Eindopdracht

- Maak een programma dat twee verschillende plaatjes gebruikt

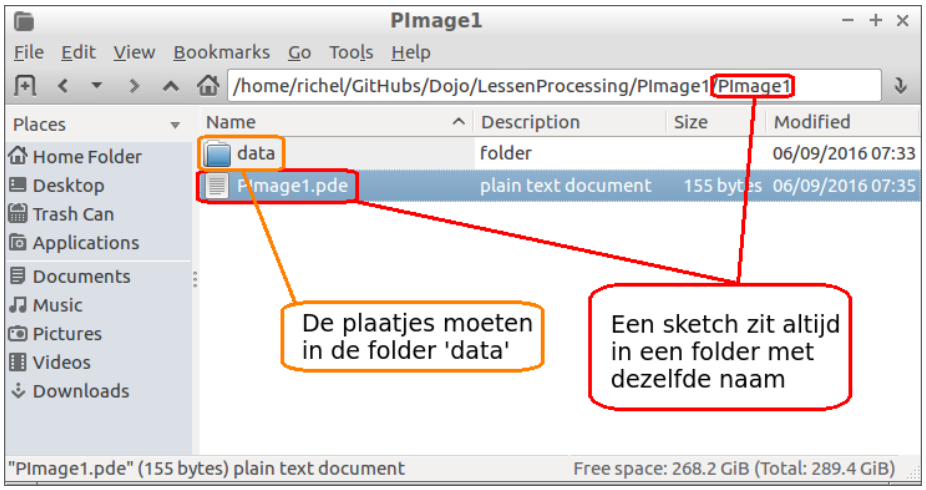


Figure 31: Folderstructuur.png

Functies les 1: Tekening

In deze les gaan we leren wat een functie is en waarom het handig is om functies te gebruiken. We doen dat met een mooie tekening van een schaap, bij dag of nacht.

of

Code

Dit is de code voor de tekening, als je de muis indrukt wordt het dag.

```
void setup() {
  size(500, 500);
}

void draw() {
  if (mousePressed) {
    //Dag
    background(128, 128, 255);
    //Teken grond
    fill(0, 255, 0);
    rect(0, 250, 500, 250);
    //Teken schaap
    line(50, 200, 50, 250);
    line(70, 200, 70, 250);
    line(100, 200, 100, 250);
```

```
    line(120, 200, 120, 250);
    fill(255);
    ellipse(80, 200, 100, 50);
    ellipse(120, 180, 30, 30);
  } else {
    //Nacht
    background(0, 0, 64);
    //Teken grond
    fill(0, 255, 0);
    rect(0, 250, 500, 250);
    //Teken schaap
    line(50, 200, 50, 250);
    line(70, 200, 70, 250);
    line(100, 200, 100, 250);
    line(120, 200, 120, 250);
    fill(255);
    ellipse(80, 200, 100, 50);
    ellipse(120, 180, 30, 30);
  }
}
```

Je ziet dat een groot gedeelte van de code er twee keer staat. Dat is het stukje met de grond en het schaap.

Dit stukje dus:

```
//Teken grond
fill(0, 255, 0);
rect(0, 250, 500, 250);
//Teken schaap
line(50, 200, 50, 250);
line(70, 200, 70, 250);
line(100, 200, 100, 250);
line(120, 200, 120, 250);
fill(255);
ellipse(80, 200, 100, 50);
ellipse(120, 180, 30, 30);
```

Functies

We kunnen bij deze code goed een functie gebruiken. Een functie kan er zo uit zien:

```
void naamVanDeFunctie(){
  //doe iets
}
```

- `void` betekent dat deze functie niets terug geeft
- `naamVanDeFunctie` is naam van de functie
- `()` tussen deze haakjes kan je argumenten zetten, dat doen we nu niet. Deze functie heeft dus geen argumenten!
- `//doe iets` hier kan je zetten wat de functie moet doen, deze code wordt uitgevoerd elke keer als de functie aangeroepen wordt

Om onze tekening code korter te maken kunnen we deze functie maken:

```
void tekenGrondEnSchaap(){
  //Teken grond
  fill(0, 255, 0);
  rect(0, 250, 500, 250);
  //Teken schaap
  line(50, 200, 50, 250);
  line(70, 200, 70, 250);
  line(100, 200, 100, 250);
  line(120, 200, 120, 250);
  fill(255);
  ellipse(80, 200, 100, 50);
  ellipse(120, 180, 30, 30);
}
```

De code wordt dan:

```
void setup() {
  size(500, 500);
}

void draw() {
  if (mousePressed) {
    //Dag
    background(128, 128, 255);
    tekenGrondEnSchaap();
  } else {
    //Nacht
    background(0, 0, 64);
    tekenGrondEnSchaap();
  }
}

void tekenGrondEnSchaap() {
  //Teken grond
  fill(0, 255, 0);
  rect(0, 250, 500, 250);
  //Teken schaap
  line(50, 200, 50, 250);
  line(70, 200, 70, 250);
```

```
  line(100, 200, 100, 250);
  line(120, 200, 120, 250);
  fill(255);
  ellipse(80, 200, 100, 50);
  ellipse(120, 180, 30, 30);
}
```

Je kan zien dat we de functie `tekenGrondEnSchaap` aanroepen met de regel `tekenGrondEnSchaap();`

Omdat we een functie gebruiken is onze code een stuk korter geworden. Ook is de code een stuk overzichtelijker en leesbaarder geworden, het lijkt bijna Nederlands!

Opdrachten

1. Kopieer de code en deel de functie 'tekenGrondEnSchaap' op in twee functies; 'tekenGrond' en 'tekenSchaap'
2. Teken een zon als het dag is
3. Teken een maan als het nacht is
4. Maak nu de functies 'tekenDag' en 'tekenNacht' en zorg dat je de kleur van de lucht en de zon of maan in die functies tekent
5. Check of 'void draw' er nu zo uit ziet:

```
void draw() {
  if (mousePressed) {
    //Dag
    tekenDag()
    tekenGrond();
    tekenSchaap();
  } else {
    //Nacht
    tekenNacht();
    tekenGrond()
    tekenSchaap();
  }
}
```

Functies les 2: Schaapkleur

Als je nog niet weet wat functies zijn, doe dan eerst deze les

In deze les gaan we kijken hoe een functie met argumenten werkt. Dat doen we door een schaap te tekenen.

We beginnen met deze code:

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

void draw() {
    //Teken lucht
    background(128, 128, 255);
    //Teken grond
    fill(0, 255, 0);
    noStroke();
    rect(0, 128, 256, 128);
    //Teken schaap
    stroke(255);
    line(50, 100, 50, 150);
    line(70, 100, 70, 150);
    line(100, 100, 100, 150);
    line(120, 100, 120, 150);
    fill(255);
    ellipse(80, 100, 100, 50);
    ellipse(120, 80, 30, 30);
}
```

Deze code tekent dit:

Stel dat ik de kleur van het schaap wil veranderen in grijs of zwart. Dat kan natuurlijk door elke keer de regels `stroke(255)` en `fill(255)` te veranderen, maar we kunnen het ook met een functie doen.

Eerst gaan we het schaap tekenen in de functie `void tekenSchaap()` die er zo uit ziet.

```
void tekenSchaap() {
  //Teken schaap
  stroke(255);
  line(50, 100, 50, 150);
  line(70, 100, 70, 150);
  line(100, 100, 100, 150);
  line(120, 100, 120, 150);
  fill(255);
  ellipse(80, 100, 100, 50);
  ellipse(120, 80, 30, 30);
}
```

Onze `void draw()` ziet er nu zo uit:

```
void draw() {
    //Teken lucht
    background(128, 128, 255);
    //Teken grond
```

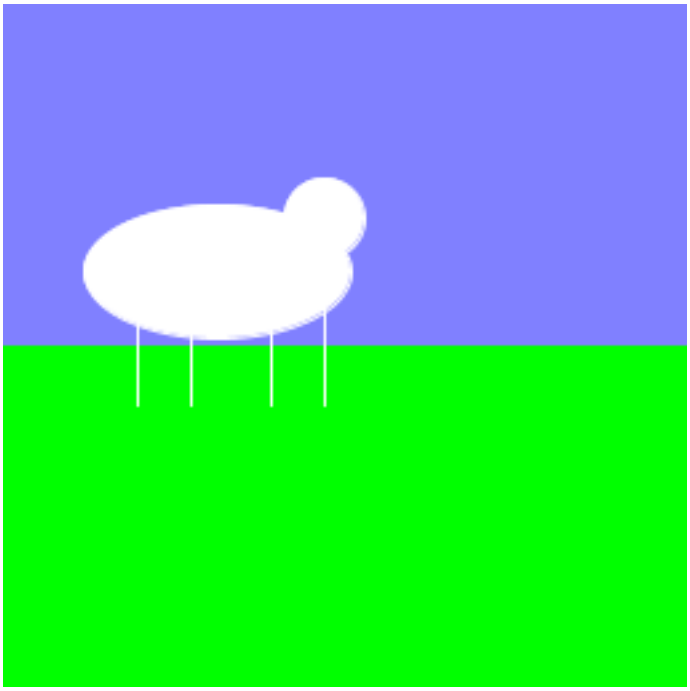


Figure 32: Wit Schaap

```
    fill(0, 255, 0);
    noStroke();
    rect(0, 128, 256, 128);
    //Teken schaap
    tekenSchaap();
}
```

Argumenten

Nu gaan we de kleur van het schaap aanpasbaar maken, dat doen we door een argument toe te voegen aan de `tekenSchaap()` functie.

Eerst veranderen we de regel

```
void tekenSchaap() {
```

in

```
void tekenSchaap(float kleur) {
```

`float kleur` betekend dat als we de functie aanroepen we er een getal in kunnen stoppen die wordt opgeslagen in het getal `kleur`

Nu kunnen we dus het getal `kleur` ook in de rest van de functie gebruiken, de `tekenSchaap()` functie ziet er dan zo uit:

```
void tekenSchaap(float kleur) {
  //Teken schaap
  stroke(kleur);
  line(50, 100, 50, 150);
  line(70, 100, 70, 150);
  line(100, 100, 100, 150);
  line(120, 100, 120, 150);
  fill(kleur);
  ellipse(80, 100, 100, 50);
  ellipse(120, 80, 30, 30);
}
```

Maar om de functie nu ook goed te gebruiken moeten we in `void draw()` de regel

```
tekenSchaap();
```

veranderen in

```
tekenSchaap(255);
```

Nu tekent het programma weer een wit schaap! Maar met deze code kunnen we heel makkelijk de kleur veranderen.

Opdrachten

- Verander het argument in `tekenSchaap` zodat het schaap zwart is.
- Verander het argument in `tekenSchaap` zodat het schaap grijs is.
- Wat denk je dat er gebeurt als je `mouseX` invult als argument? Probeer het om je hypthese te controleren!

Functies les 3: Poten

Als je nog niet weet wat functies zijn, doe dan eerst deze les

Als je nog niet weet hoe functies met argumenten werken, doe dan eerst deze les

In deze les gaan we kijken naar functies met een return type. Dat doen we door poten van een aantal schapen te tellen.

We beginnen met deze code:

```
int schapen;

void setup(){
  size(256, 256);
  println("Potenberekenaar!");
  background(25, 160, 25);

  schapen = 2;
  println("Voor " + schapen + " schapen, zijn er " + schapen * 4 + " poten!");
  schapen = 3;
  println("Voor " + schapen + " schapen, zijn er " + schapen * 4 + " poten!");
  schapen = 5;
  println("Voor " + schapen + " schapen, zijn er " + schapen * 4 + " poten!");

}

void draw(){

}
```

Deze code print dit naar de console:

```
Potenberekenaar!
Voor 2 schapen, zijn er 8 poten!
Voor 3 schapen, zijn er 12 poten!
Voor 5 schapen, zijn er 20 poten!
```

Er staat elke keer `schapen * 4`, omdat elk schaap 4 poten heeft. Maar dat staat niet heel duidelijk in de code.

Het zou een stuk leesbaarder zijn als er `aantalPoten(schapen)` stond. En dat kan!

Maar dan moeten we wel eerst de `aantalPoten` functie maken, die het aantal schapen ontvangt en het aantal poten teruggeeft.

Als we een functie willen maken die iets teruggeeft moeten we in plaats van `void` het return type gebruiken. In dit geval is dat `int`, omdat het aantal poten altijd een heel getal is.

De functie wordt dan:

```
int aantalPoten(int aantalSchapen){
  return aantalSchapen * 4;
}
```

In de regel `return aantalSchapen * 4;` geven we `aantalSchapen * 4` terug. `return` betekent dus `geef terug`!

Opdrachten

- Kopieër de code van het programma aan het begin en pas het aan zodat het de `aantalPoten` functie gebruikt wordt
- Bedenk een manier om iets op het scherm te tekenen bij dit programma
- Maak een functie die het aantal oren teruggeeft in plaats van het aantal poten
- Maak een functie die het aantal schapen teruggeeft als je er het aantal poten in stopt