

Processing

Voorwoord

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.

(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 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.

Processing opstarten op cursuslaptop

Met een terminal kun je veel dingen doen, die soms niet met een normaal programma gedaan kunnen worden. Op de cursus start je Processing vanuit de terminal.

Processing starten

Start een Terminal. Dit kan soms met Win+T, CTRL+ALT+T, of deze te vinden in de menuutjes, of te zoeken op het woord **Terminal**

Ga naar de folder waar Processing in staat. Hiervoor is het commando **cd**. De afkorting **cd** staat voor ‘Change Directory’. “Change Directory” is Engels voor ‘Verander van folder’.

Zo ga je naar de folder waar Processing instaat:

```
cd Programs/processing-3.1.1
```

Dit hoeft je niet zo te typen! Een terminal kan je woorden aanvullen als je op Tab drukt. Vaak is het volgende typen voldoende:

```
cd Progr[TAB]/pr[TAB]
```

The screenshot shows the Processing 3.2.3 IDE interface. The title bar reads "Voorwoord | Processing 3.2.3". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The toolbar has a play/pause button and a Java dropdown. The code editor window contains the following text:

```
Voorwoord
1 *****
2      PROCESSING
3 *****
4
5 Dit is het Processing boek
6 van de Dojo.
7
8 Processing is een programmeertaal.
9
10 Dit boek leert je die taal.
11
12 Over dit boek
13 -----
14
15 Dit boek heeft een CC-BY-NC-SA
16 licentie.
17
18 (C) Dojo Groningen 2016 CC-BY-NC-SA
19 */
20
```

At the bottom of the code editor, there are two banners: "DE JONGE ONDERZOEKERS" with the tagline "Kijk, doe, ontdek!" and "CODESTARTER". The status bar at the bottom says "Done saving." and has tabs for "Console" and "Errors".

Figure 1: Voorblad



Figure 2: Het logo van De Jonge Onderzoekers



Figure 3: Het logo van Codestarter



Figure 4: De licensie van dit boek

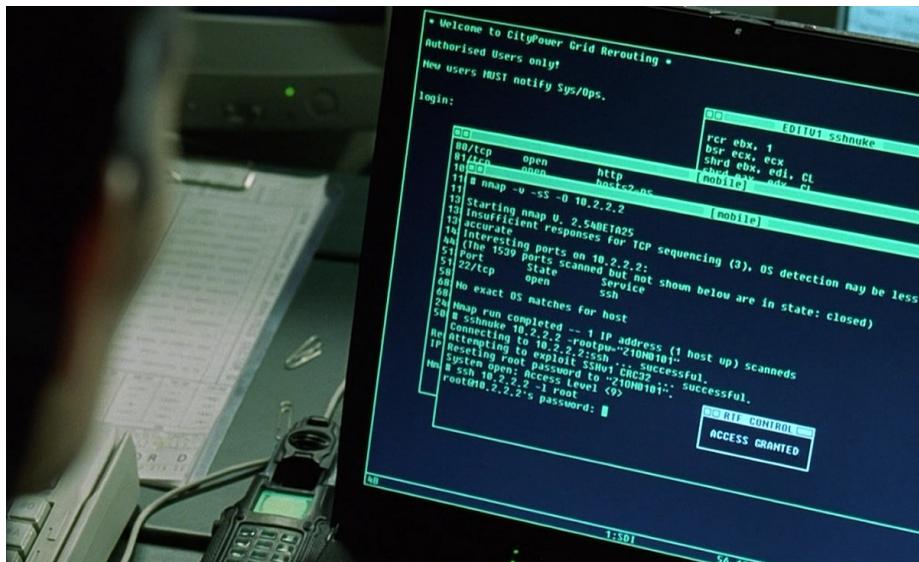


Figure 5: Hackers werken met een terminal



Figure 6: Een terminal

Dit werkt ook als een andere versie van Processing op de computer staat :-)

Nu je in de juiste folder bent, start Processing:

`./processing`

De `./` betekent ‘Start hier’

Ook hier is Tab nuttig:

`./p[TAB]`

Je mag de terminal nu sluiten.

Opdrachten

- Start Processing

Een Mooi Programma

Processing is een programmeertaal ontwikkeld voor kunstenaars en erg geschikt om games en mooie dingen mee te maken.

In deze les gaan we leren

- hoe we Processing opstarten
- hoe je code naar Processing kopieert
- hoe je het programma start

Zo ziet het programma eruit:

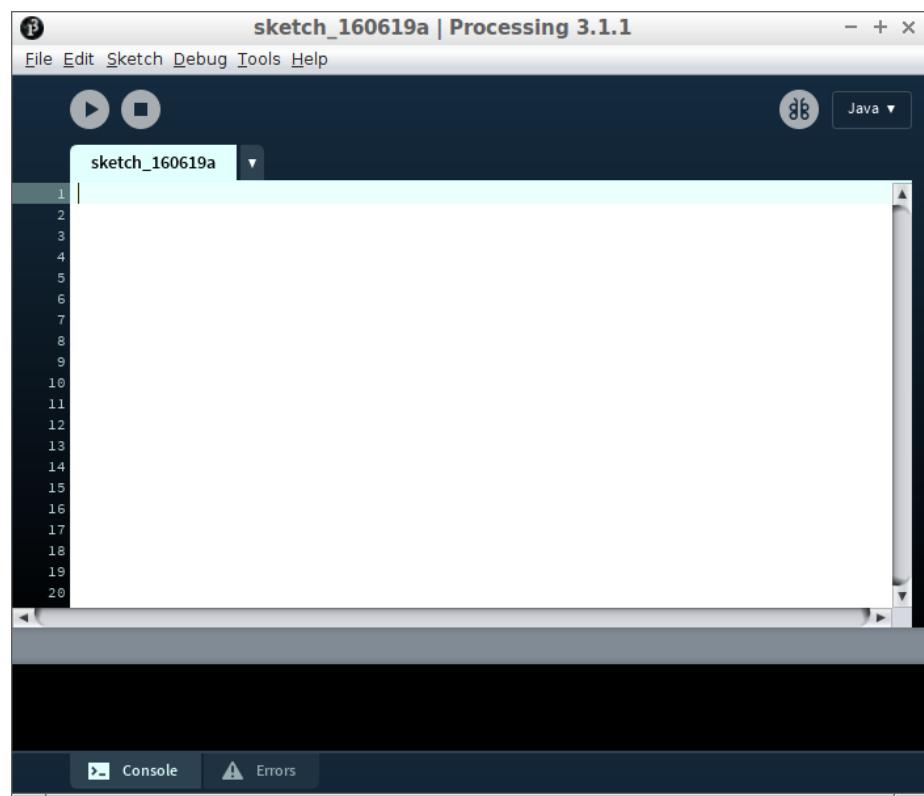


Figure 7: Processing zonder code



Figure 8: EenMooiProgramma

Wat je nodig hebt

Je moet Processing op kunnen starten. Hoe dat moet, hangt af van het besturingssysteem:

- Processing opstarten op cursus laptop
- Processing installeren op eigen laptop met GNU/Linux
- Processing installeren op eigen laptop met Windows

Code kopieeren

Processing begint met een leeg programma zonder code:

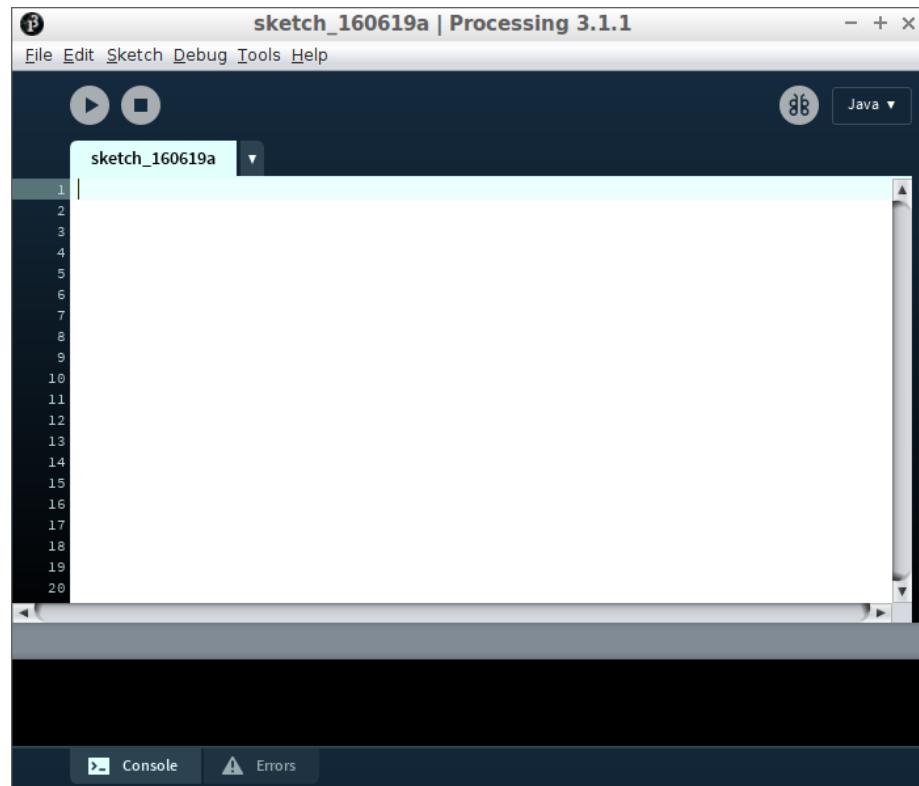


Figure 9: Processing zonder code

Dit is de programmeercode die we gaan gebruiken:

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

```

}

void draw()
{
    fill(mouseX, mouseY, mouseX + mouseY);
    ellipse(mouseX, mouseY, 50, 50);
    fill(mouseY, mouseX, 255);
    ellipse(mouseY, mouseX, 50, 50);
}

```

Wat de code precies doet, leggen we later uit. Voor nu is het genoeg te weten dat het iets moois doet.

Om code te kopieeren gebruik je sneltoetsen:

- SHIFT + pijltjes: selecteren
- CTRL + A: alles selecteren
- CTRL + C: kopiëren van selectie
- CTRL + X: knippen van selectie
- CTRL + V: plakken van selectie
- Start Processing
- Kopieer deze code naar Processing

Programma uitvoeren

- Klik op de ‘Run’ knop

Als het goed is, zie je nu het programma!

Sneltoetsen oefenen

- Werk met iemand samen. Hussel de code van de andere door de war, door deze te knippen/kopiëren en te plakken. Repareer dan de code op je eigen computer

Point

Processing is een programmeertaal ontwikkeld voor kunstenaars en erg geschikt om games en mooie dingen mee te maken.

In deze les gaan we leren

The screenshot shows the Processing 3.1.1 software interface. The title bar reads "sketch_160619a | Processing 3.1.1". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. On the right side of the toolbar, there are buttons for "Java" and "JavaScript". The main code editor window displays the following Java code:

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

The code uses the Processing API to set the window size to 256x256 pixels. In the draw() function, it fills the background with the sum of the mouse coordinates and then draws two overlapping ellipses: one centered at the mouse position with a radius of 50 pixels, and another centered at the mouse position with a radius of 50 pixels, filled with white (#FFFF00).

Figure 10: Processing met code

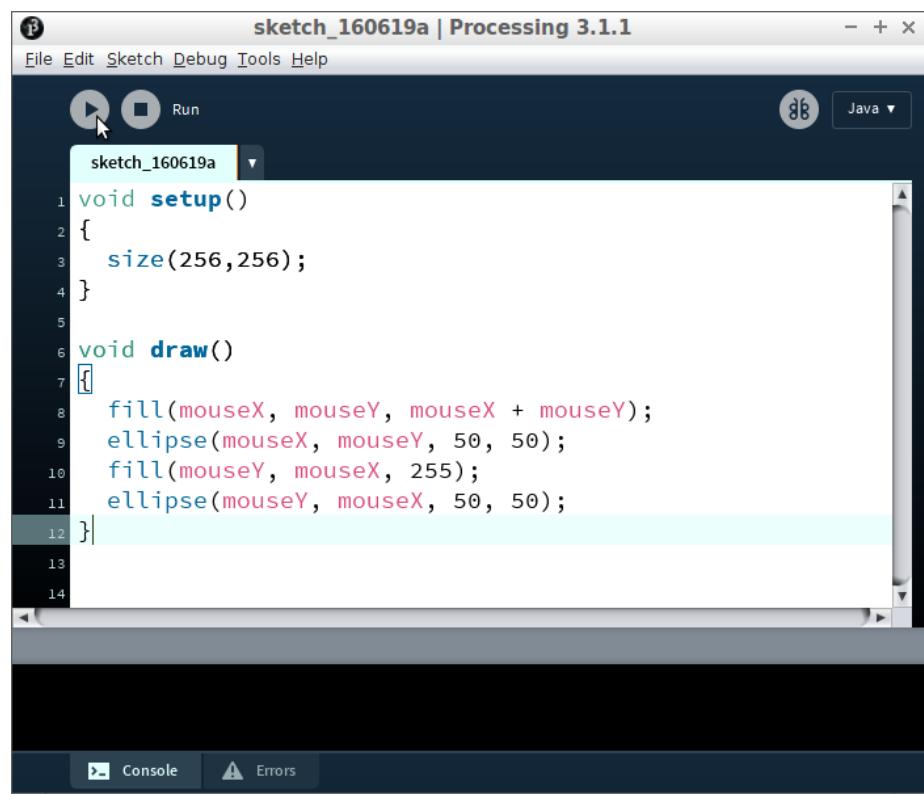


Figure 11: De Run knop

- wat pixels zijn
- hoe de pixels op een beeldscherm zitten
- hoe je puntjes tekent

Zo gaat het eruit zien:

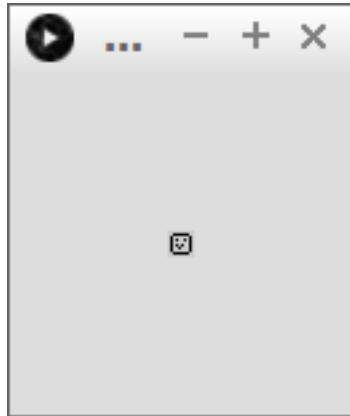


Figure 12: Point

Pixels

Pixels zijn de vierkantjes waaruit je beeldscherm is opgebouwd. Je schermresolutie is het aantal pixels dat je scherm breed en hoog is. Hoe meer pixels je scherm heeft, hoe scherper je beeldscherm eruit ziet

Vroeger hadden computers een lage resolutie. Daarom zien oude spellen er wat blokkiger uit:

Vragen

- Wat is de resolutie van jouw beeldscherm?
- Als een beeldscherm een resolutie heeft van 320 bij 200 pixels, hoeveel pixels heeft deze dan?

Hoe zitten pixels op je beeldscherm

Elke pixel op je beeldscherm heeft een soort postcode. Deze postcode noemen we een coordinaat. Een coordinaat bestaat uit twee getallen. Dit zijn de coordinaten van alle pixels van een beeldscherm van vijf bij vijf pixels:

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)

Figure 13: Pixel coordinaat

Je ziet dat de laagste coördinaat $(0,0)$ (zeg: ‘nul komma nul’) is. $(0,0)$ zit in de linkerbovenhoek van je beeldscherm. Dan zie je dat het eerste getal is hoeveel pixels naar recht hiervan is. Zo zit $(1,0)$ rechts van $(0,0)$. Het tweede getal is hoeveel pixels je onder $(0,0)$ zit. Zo zit $(0,1)$ onder $(0,0)$.

Op deze manier kun je elke pixel op je beeldscherm vinden.

Vragen

- Welke coördinaten heb je nodig om een horizontale lijn te tekenen, die door $(0,0)$ gaat en drie pixels lang is?
- Welke coördinaten heb je nodig om een verticale lijn te tekenen, die door $(1,0)$ gaat en drie pixels hoog is?
- Welke coördinaten heb je nodig om een diagonale lijn te tekenen, die door $(0,0)$ gaat en drie pixels lang is?
- Welke coördinaten heb je nodig om een vierkantje te tekenen, die door $(0,0)$ gaat, twee pixels breed en twee pixels breed is?

Puntjes tekenen

Dit is de programmeercode die je nodig hebt:

```
void setup()
{
    size(10,10);
}

void draw()
{
    // 0123456789
    // 0 .....
    // 1 ..XXXXXX..
    // 2 .X....X.
    // 3 .X.X..X.X.
    // 4 .X.....X.
    // 5 .X.X.X..X.
    // 6 .X..X....X.
    // 7 .X.....X.
    // 8 ..XXXXXX..
    // 9 .....

    // Bovenkant hoofd
    point(2,1);
    point(3,1);
```

```

    point(4,1);
    point(5,1);
    point(6,1);
    point(7,1);

    // Rechterkant hoofd
    point(8,2);
    point(8,3);
    point(8,4);
    point(8,5);
    point(8,6);
    point(8,7);

    // Onderkant hoofd
    point(2,8);
    point(3,8);
    point(4,8);
    point(5,8);
    point(6,8);
    point(7,8);

    // Linkerkant hoofd
    point(1,2);
    point(1,3);
    point(1,4);
    point(1,5);
    point(1,6);
    point(1,7);

    // Ogen
    point(3,3);
    point(6,3);

    // Mond
    point(3,5);
    point(4,6);
    point(5,5);

}

```

Dit is wat de code doet:

- `void setup() {}`: de setup functie, de computer voert een keer alles tussen de accolades uit
- `size(10, 10)`: maak een scherm van 10 pixels breed en 10 pixels hoog
- `;`: de puntkomma geeft in Processing het einde aan van een zin. Dit is ongeveer hetzelfde met een punt in schrijftaal.

- `void draw() {}`: de draw functie, de computer voert steeds alles tussen de accolades uit
- `//`: commentaar: de rest van de regel wordt niet gelezen door de computer. Hier kun je dingen neerzetten die speciaal bedoelt zijn voor mensen, zoals jezelf. In dit programma zie je dat het poppetje is getekent in commentaar, met de coordinaten aan de zijkant. Ook zie je dat de programmeur zegt wat er getekend wordt: zo kun je de fouten sneller vinden
- `point(2,1)`: teken een puntje op coordinaat (2,1).

Vragen

- Start Processing. Kopieer deze code en run het programma
- Laat de smiley een pixel breder lachen
- Draai de lach verticaal om, zodat de smiley sip gaat kijken
- Geef de smiley punk haar door drie pixels bovenop het hoofd erbij te tekenen
- Geef de smiley een sik van een pixel

Line

Zonder lijnen kun je bijna geen games maken. Een van de allereerste successgames, Asteroids, bestond vooral uit lijnen:

Je kunt een lijn tekenen met een boel puntjes, maar de `line` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je lijnen tekent

Zo gaat het eruit zien:

Kun je nog geen puntjes tekenen? Ga dan naar de les waarin je puntjes leert tekenen

Lijnen

Een lijn bestaat uit pixels. Om een lijn te tekenen, moet je een beginpixel en eindpixel kiezen. Processing tekent dan zelf de pixels ertussenin.

Om in Processing een lijn te tekenen, gebruik je de functie `line`. De functie `line` heeft vier getallen nodig. Deze vier getallen zijn twee coordinaten achter elkaar. De eerste twee getallen zijn de coordinaat van het ene eind van de lijn. De laatste twee getallen zijn de coordinaat van het ander eind van de lijn.



Figure 14: Asteroids

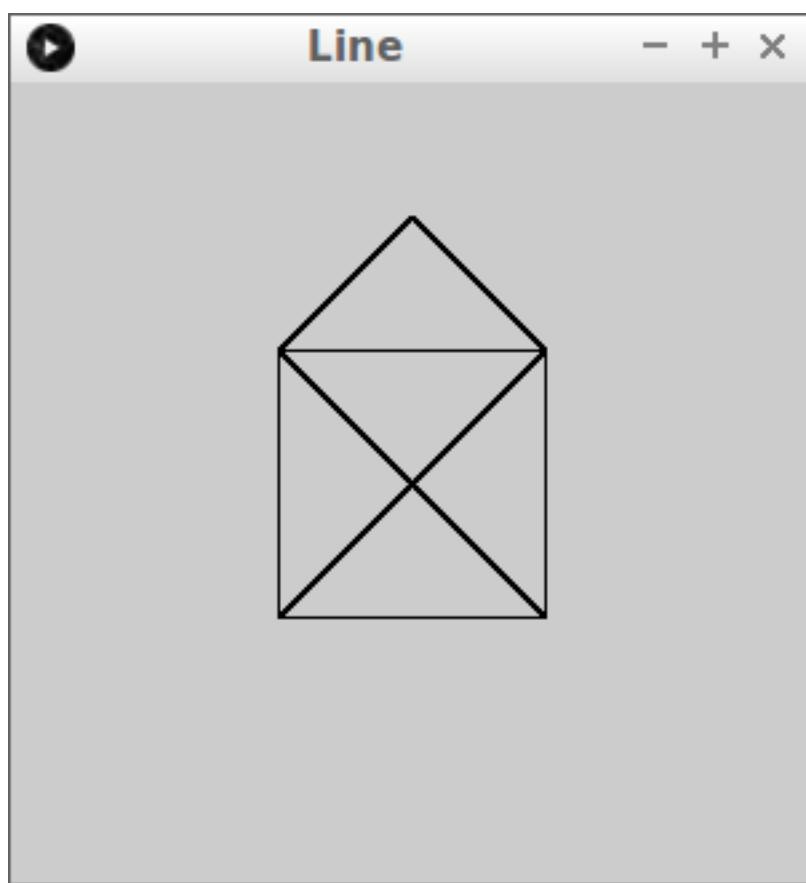


Figure 15: Line

Hier zie je een lijn die gaat van coordinaat (1,2) naar (3,2):

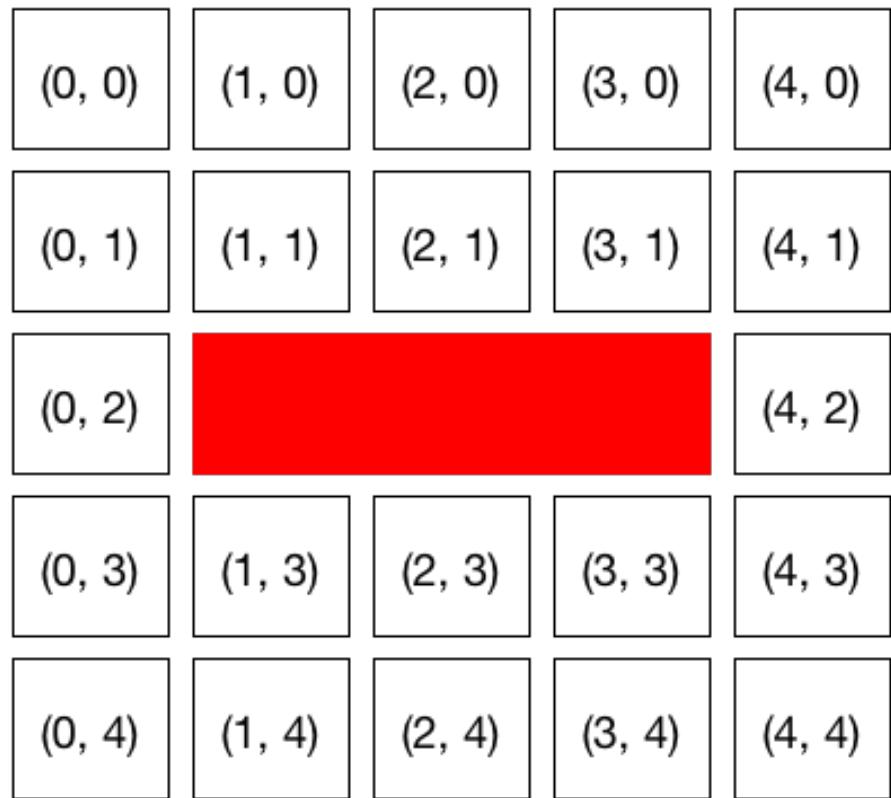


Figure 16: Lijn 1

In Processing teken je deze lijn met:

```
line(1,2,3,2);
```

De volgorde van de twee coordinaten maakt niet uit.

Lijnen kunnen ook schuin gaan.

Hier zie je een lijn die gaat van coordinaat (2,4) naar (1,1):

In Processing teken je deze lijn met:

```
line(2,4,1,1);
```

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)		(2, 1)	(3, 1)	(4, 1)
(0, 2)		(2, 2)	(3, 2)	(4, 2)
(0, 3)	(1, 3)		(3, 3)	(4, 3)
(0, 4)	(1, 4)		(3, 4)	(4, 4)

Figure 17: Lijn 2



(1, 0)	(2, 0)	(3, 0)	(4, 0)
(1, 1)	(2, 1)	(3, 1)	(4, 1)
(1, 2)	(2, 2)	(3, 2)	(4, 2)
(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)

Figure 18: Lijn 3

Vragen

- 1. Hierboven staat een lijn. Wat zijn de begin- en eindcoordinaat van die lijn?

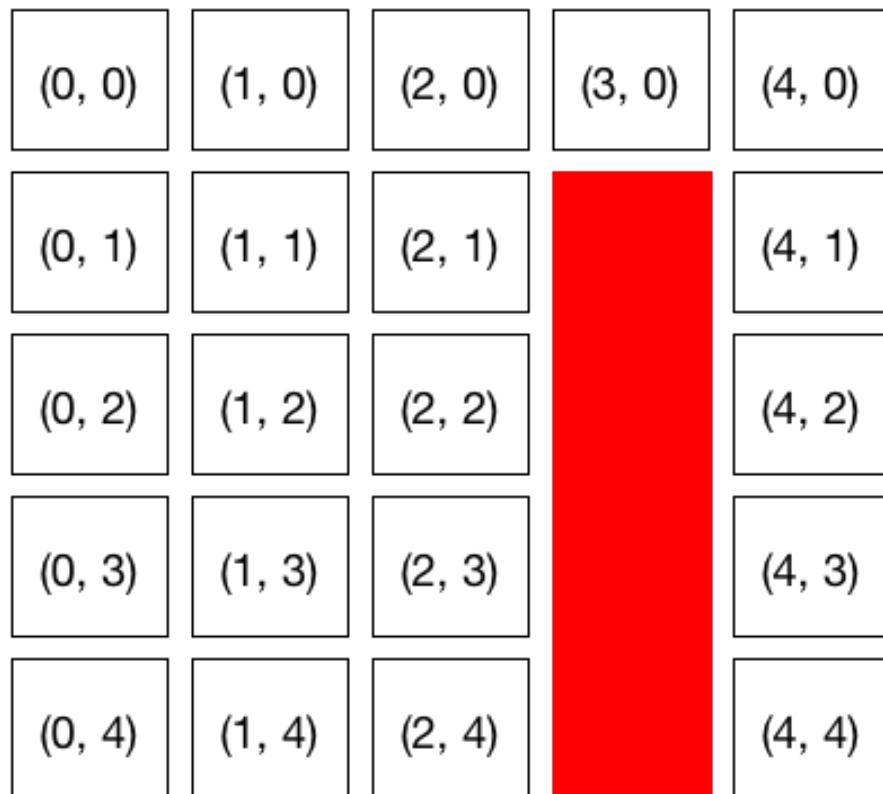


Figure 19: Lijn 4

- 2. Hierboven staat een lijn. Wat zijn de begin- en eindcoordinaat van die lijn?
- 3. Hierboven staat een lijn. Wat zijn de begin- en eindcoordinaat van die lijn?
- 4. Een lijn gaat van coordinaat (0,0) naar (10,0). In welke richting gaat de lijn? Wat is het Processing commando?
- 5. Een lijn gaat van coordinaat (0,0) naar (0,10). In welke richting gaat de lijn? Wat is het Processing commando?
- 6. Een lijn gaat van coordinaat (0,0) naar (10,10). In welke richting gaat de lijn? Wat is het Processing commando?

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)
(0, 1)	(1, 1)		(3, 1)	(4, 1)
(0, 2)		(2, 2)	(3, 2)	(4, 2)
	(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)

Figure 20: Lijn 5

- 7. Een lijn gaat van coordinaat (30,20) naar (20,20). In welke richting gaat de lijn? Wat is het Processing commando?
- 8. Een lijn gaat van coordinaat (10,20) 20 pixels naar rechts. Welke coordinaat heeft het eindpunt? Wat is het Processing commando?
- 9. Een lijn gaat van coordinaat (10,30) 10 pixels naar rechtsomhoog. Welke coordinaat heeft het eindpunt? Wat is het Processing commando?

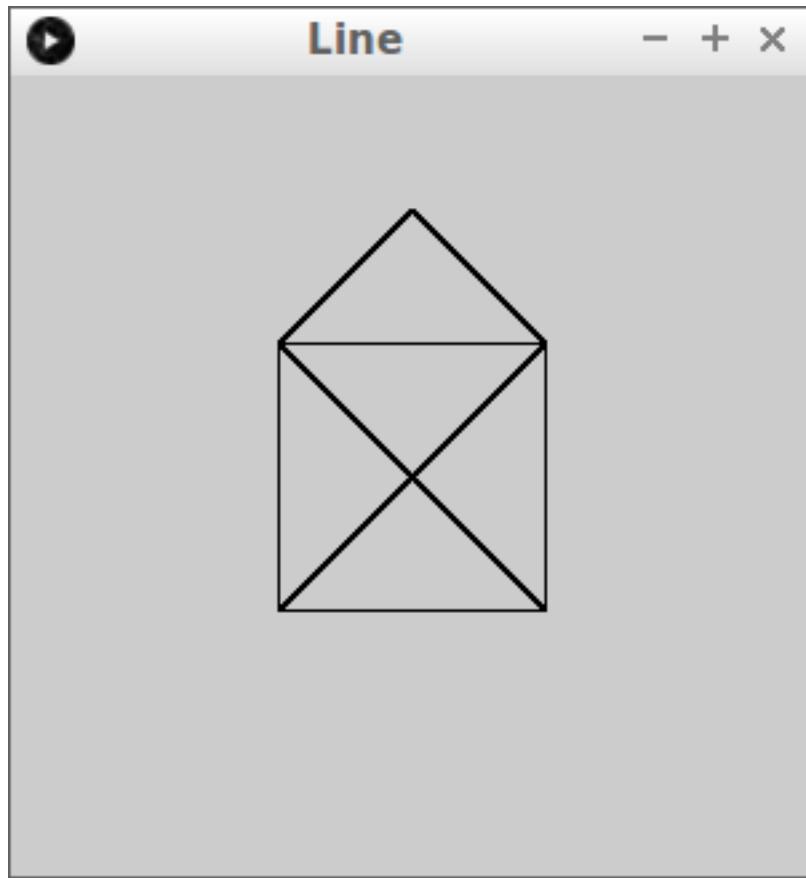


Figure 21: Line

- 10. Hierboven staat een tekening. Maak deze tekening na in Processing

Oplossing

- 1. (0,0) en (0,3)

- 2. (3,1) en (3,4)
- 3. (2,1) en (0,3)
- 4. Van links naar rechts/horizontaal. `line(0,0,10,0)`
- 5. Van onder naar boven/verticaal. `line(0,0,0,10)`
- 6. Schuin/diagonaal. `line(0,0,0,10)`
- 7. Van rechts naar links/horizontaal. `line(30,20,20,20)`
- 8. (30,20). `line(10,20,30,20)`
- 9. (20,20). `line(10,30,20,20)`
- 10. Zie hieronder:

```

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

void draw()
{
    //
    //      a
    //      / \
    //      e---b
    //      | \ / |
    //      |   X  |
    //      | / \ \
    //      d---c
    //
    // a: (150, 50)
    // b: (200,100)
    // c: (200,200)
    // d: (100,200)
    // e: (100,100)

    //Van a naar b
    line(150,50,200,100);
    //Van b naar c
    line(200,100,200,200);
    //Van c naar d
    line(200,200,100,200);
    //Van d naar e
    line(100,200,100,100);
    //Van e naar a
    line(100,100,150,50);
}

```

```
//Van b naar d  
line(200,100,100,200);  
//Van b naar e  
line(200,100,100,100);  
//Van c naar e  
line(200,200,100,100);  
}  
}
```

backGround

Zonder kleur zien games er minder mooi uit.

Een van de allereerste games met kleur heette Moria:



Figure 22: Moria

Om goed kleuren te kunnen gebruiken, moeten we leren hoe kleuren werken.

In deze les gaan we leren

- hoe kleuren werken

Zo gaat het eruit zien:

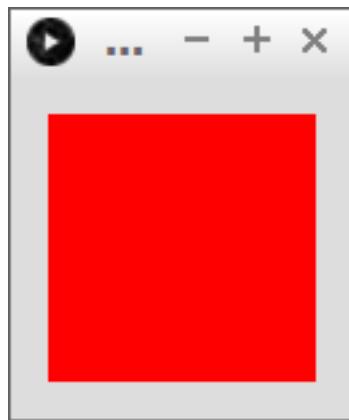


Figure 23: background

Je hoeft maar weinig te weten, behalve hoe je een mooi programma maakt

Kleuren

Als je goed naar een beeldscherm kijkt, zie je dat elke pixel uit drie lampjes bestaat:

De lampjes hebben de kleuren rood, groen en blauw.

Omdat de lampjes zo klein zijn, ziet ons oog van afstand de menging van deze drie lampjes. Zo zien de lampjes rood en groen er samen uit als geel. Hier zie je hoe de kleuren mengen:

Om wit te krijgen, heb je alledrie de kleuren nodig.

Vragen

- 1. Welke drie kleuren lampjes heeft een pixel?
- 2. Met welke lampjes samen maak je geel?
- 3. Met welke lampjes samen maak je cyaan/lichtblauw?
- 4. Met welke lampjes samen maak je magenta/paars?

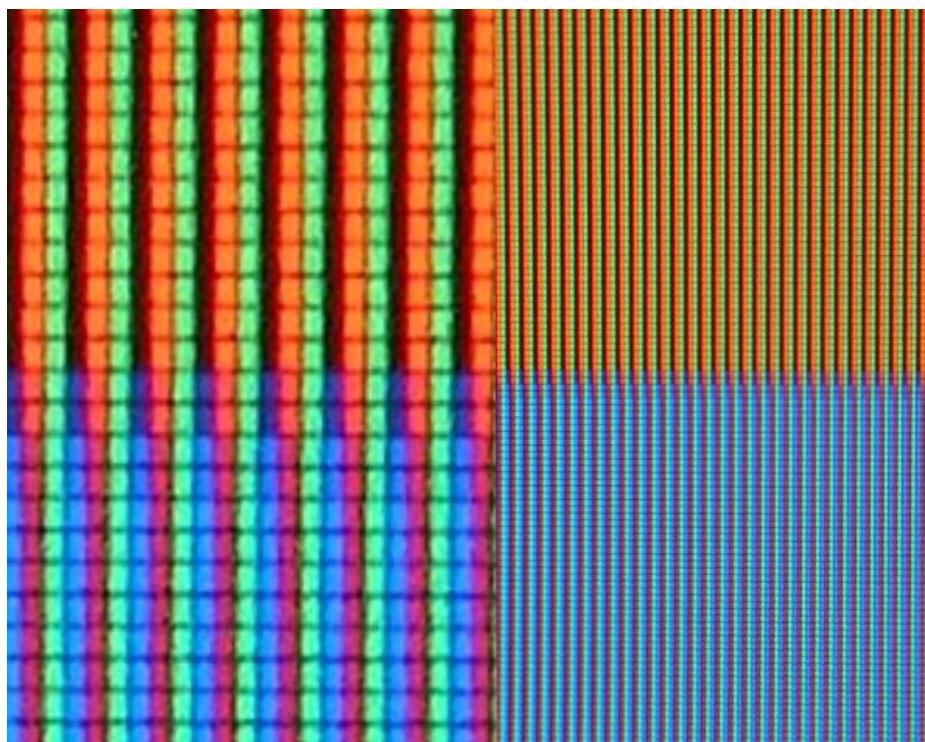


Figure 24: RGB pixels

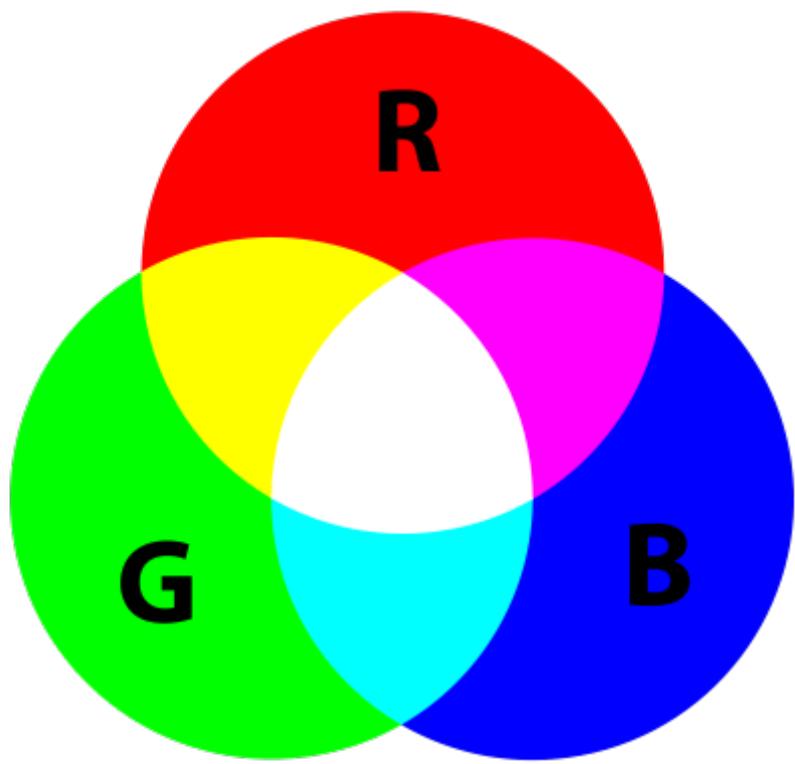


Figure 25: Additieve kleuren

- 5. Met welke lampjes samen maak je wit?
- 6. Met welke lampjes samen maak je zwart?
- 7. Met welke lampjes samen maak je grijs?
- 8. Met welke lampjes samen maak je oranje?

Antwoorden

- 1. Rood, groen en blauw
- 2. Rood en groen
- 3. Groen en blauw
- 4. Rood en blauw
- 5. Rood en groen en blauw
- 6. Met geen lampjes: als alle lampjes uit zijn, is het zwart
- 7. Met rood en groen en blauw, maar dan moeten ze niet op hun hardst branden
- 8. Met rood op z'n hardst en groen op halve kracht

background

In Processing is er een functie om de achtergrond een kleur te geven. Deze functie heet **background**. **background** is Engels voor ‘achtergrond’. **background** is een functie die drie getallen nodig heeft. Deze drie getallen bepalen hoe hard de rode, groene en blauwe lampjes gaan schijnen. Deze drie getallen noemen we de RGB waarde. Met het getal nul zeg je dat een lampje uit staat. Met het getal 255 zeg je dat een lampje aan staat. Met getallen tussen nul en 255 kun je het lampje ertussenin laten branden.

Met deze Processing code krijg je een rode achtergrond:

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

void draw()
{
    background(255,0,0);
}
```

Opdracht

- 1. Kopieer deze code in Processing en start de code
- 2. Wat is een RGB waarde?
- 3. Wat is de RGB waarde van groen? Maak in Processing een groene achtergrond
- 4. Wat is de RGB waarde van blauw? Maak in Processing een blauwe achtergrond
- 5. Wat is de RGB waarde van geel? Maak in Processing een gele achtergrond
- 6. Wat is de RGB waarde van cyaan/lichtblauw? Maak in Processing een cyane/lichtblauwe achtergrond
- 7. Wat is de RGB waarde van magenta/paars? Maak in Processing een magenta/paarse achtergrond
- 8. Wat is de RGB waarde van wit? Maak in Processing een witte achtergrond
- 9. Wat is de RGB waarde van zwart? Maak in Processing een zwarte achtergrond
- 10. Wat is de RGB waarde van grijs? Maak in Processing een grijze achtergrond
- 11. Wat is de RGB waarde van donkerrood? Maak in Processing een donkerrode achtergrond
- 12. Wat is de RGB waarde van oranje? Maak in Processing een oranje achtergrond

Oplossingen

- 1. OK
- 2. De Rood-Groen-Blauw waarde. Dit zijn drie getallen van nul tot en met 255 die bepalen hoe hard de drie kleurenlampjes branden
- 3. `background(0,255,0)`
- 4. `background(0,0,255)`
- 5. `background(255,255,0)`
- 6. `background(0,255,255)`
- 7. `background(255,255,0)`
- 8. `background(255,255,255)`

- 9. `background(0,0,0)`
- 10. `background(128,128,128)`, maar andere getallen tussen 0 en 255 zijn ook goed. Als ze maar alledrie gelijk zijn
- 11. `background(128,0,0)`, maar het eerste getal mag ook een ander getal tussen de 0 en 255 zijn
- 12. `background(255,128,0)`, maar het tweede getal mag ook een ander getal in de buurt van 128 zijn

stroke

Zonder kleur zien games er minder mooi uit.

Een van de allereerste games met kleur heette Moria:

In deze les gaan we leren

- hoe je een lijn een kleur kan geven.

Zo gaat het eruit zien:

Weet je nog niet hoe kleuren werken, ga dan naar de les background

stroke

In Processing is er een functie om lijnen een kleur te geven. Deze functie heet `stroke`. `stroke` is Engels voor ‘(penseel)streek’. `stroke` is een functie die drie getallen nodig heeft. Deze drie getallen zijn de RGB waarden.

Met deze Processing code krijg je een rode lijn:

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

void draw()
{
    stroke(255,0,0);
    line(10,20,30,40);
}
```

Met `stroke` zeg je: ‘vanaf nu wil ik deze lijnkleur’. Hieronder zie je hoe je twee groene en een blauwe lijn tekent:

```
void setup()
{
```



Figure 26: Moria

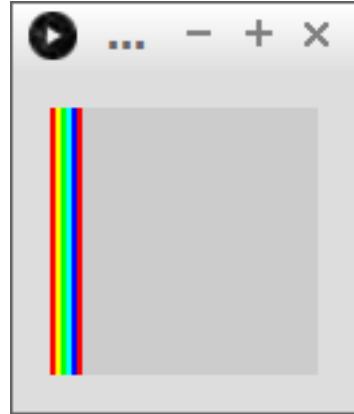


Figure 27: Stroke

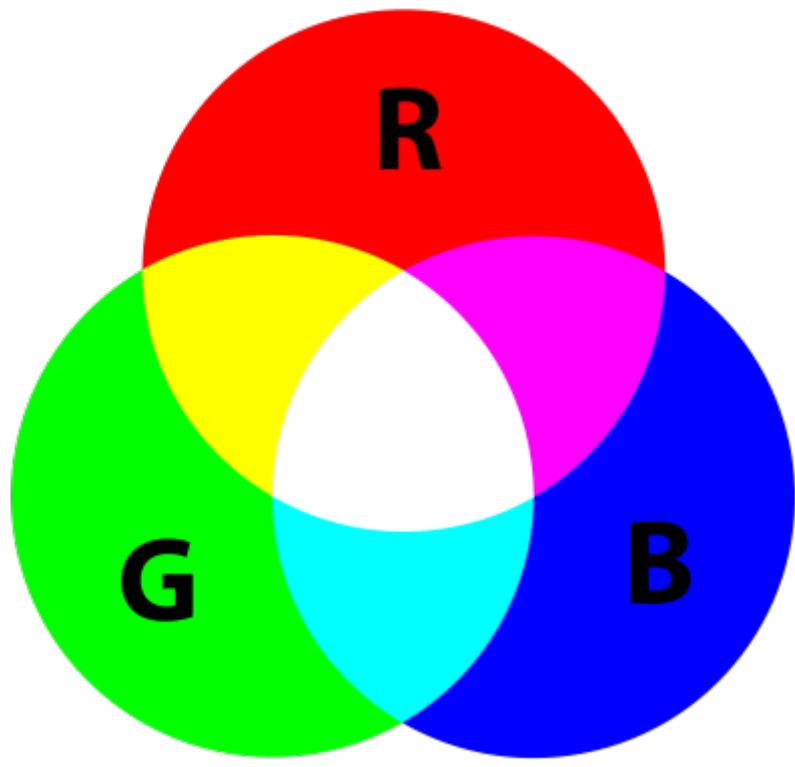


Figure 28: Kleurencirkel

```

    size(100,100);
}

void draw()
{
    stroke(0,255,0);
    line(10,20,30,40);
    line(50,60,70,80);
    stroke(0,0,255);
    line(90,10,20,30);
}

```

Opdracht

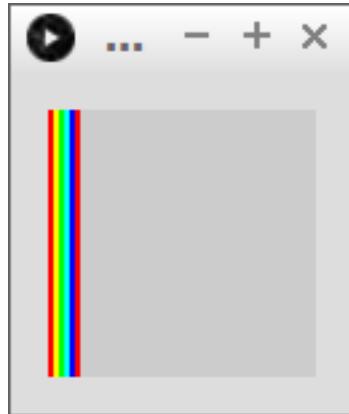


Figure 29: Stroke

- 1. Maak in Processing bovenstaande tekening na. Het venster is honderd bij honderd pixels. Elke kleur is met twee lijnen getekend. De kleuren zijn rood, geel, groen, cyaan, blauw, magenta

Oplossing

```

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

void draw()
{
    stroke(255,0,0);

```

```

line(0,0,0,100);
line(1,0,1,100);
stroke(255,255,0);
line(2,0,2,100);
line(3,0,3,100);
stroke(0,255,0);
line(4,0,4,100);
line(5,0,5,100);
stroke(0,255,255);
line(6,0,6,100);
line(7,0,7,100);
stroke(0,0,255);
line(8,0,8,100);
line(9,0,9,100);
stroke(255,0,0);
line(10,0,10,100);
line(11,0,11,100);
}

```

Rect

Vierkanten worden veel gebruikt in games.

Hier zie je een van de beroemdste games ooit:

Je kunt een vierkant tekenen met vier lijnen, maar de `rect` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je lijnen tekent

Zo gaat het eruit zien:

Kun je nog geen lijnen tekenen? Ga dan naar de les waarin je lijnen leert tekenen

Rechthoeken

Een rechthoek bestaat uit vier lijnen. Om een rechthoek te tekenen, moet je een coordinaat, breedte en hoogte geven.

Om in Processing een rechthoek te tekenen, gebruik je de functie `rect`. De functie `rect` heeft vier getallen nodig. De eerste twee getallen zijn de coordinaat van de linkerbovenhoek van de rechthoek. Het derde getal is de breedte van de rechthoek. Het vierde getal is de hoogte van de rechthoek.



Figure 30: Tetris

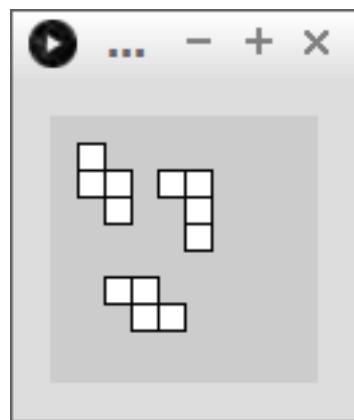


Figure 31: Rect

Hier zie je een rechthoek met coordinaat (1,2), een breedte van drie pixels en hoogte van vier pixels:

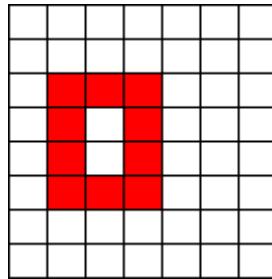


Figure 32: Rechthoek 1

In Processing teken je deze rechthoek met:

```
rect(1,2,3,4);
```

Hier is nog een rechthoek:

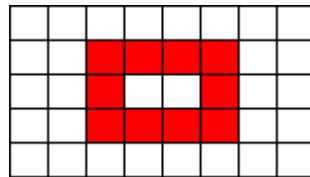


Figure 33: Rechthoek 2

De linkerbovenhoek heeft coordinaat (2,1), hij is vier pixels breed en drie pixels hoog.

Vragen

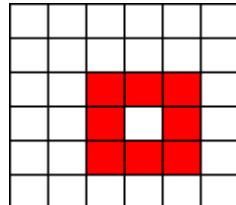


Figure 34: Rechthoek 3

- 1. Hierboven staat een rechthoek. Wat is de coordinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?

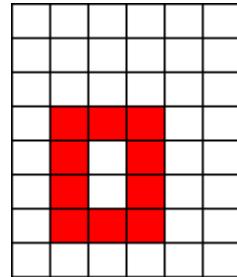


Figure 35: Rechthoek 4

- 2. Hierboven staat een rechthoek. Wat is de coordinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?

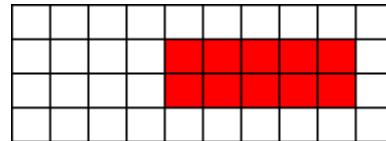


Figure 36: Rechthoek 5

- 3. Hierboven staat een rechthoek. Wat is de coordinaat van de linkerbovenhoek? Hoe breed is de rechthoek? Hoe hoog is de rechthoek?
- 4. Een rechthoek heeft als coordinaat (0,0), is twee pixels breed en drie pixels hoog. Wat is het Processing commando?
- 5. Een rechthoek heeft als coordinaat (1,2), is drie pixels breed en vier pixels hoog. Wat is het Processing commando?
- 6. Een rechthoek heeft als coordinaat (10,20), is dertig pixels breed en veertig pixels hoog. Wat is het Processing commando?
- 10. Hierboven staat een tekening. Maak deze tekening na in Processing

Oplossing

- 10. Zie hieronder:

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

void draw()
{
```

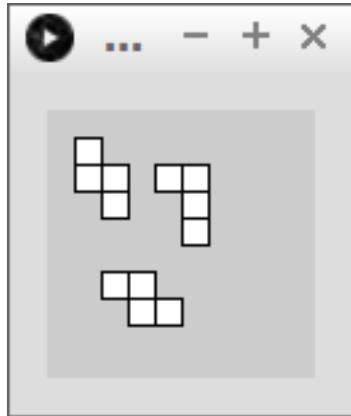


Figure 37: Rect

```
rect(10,10,10,10);
rect(10,20,10,10);
rect(20,20,10,10);
rect(20,30,10,10);

rect(40,20,10,10);
rect(50,20,10,10);
rect(50,30,10,10);
rect(50,40,10,10);

rect(20,60,10,10);
rect(30,60,10,10);
rect(30,70,10,10);
rect(40,70,10,10);
}
```

Ellipse

Cirkels en ovalen worden veel gebruikt in games.

Hier zie je een beroemde game, Bubble Bobble, dat veel met cirkels werkt:

Je kunt een ovaal tekenen met heel veel puntjes, maar de `ellipse` functie werkt gemakkelijker.

In deze les gaan we leren

- hoe je ovalen tekent

Zo gaat het eruit zien:



Figure 38: Bubble Bobble

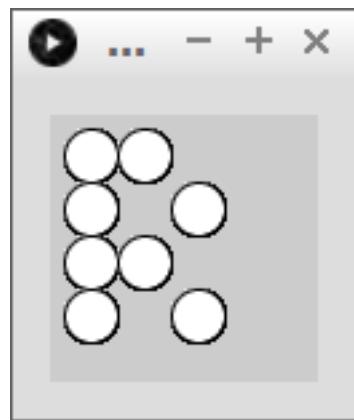


Figure 39: Ellipse

Kun je nog geen rechthoeken tekenen? Ga dan naar de les waarin je rechthoeken leert tekenen

Ovalen

Een ovaal heeft een middelpunt, breedte en hoogte. Om een ovaal te tekenen, moet je een coördinaat, breedte en hoogte geven.

Om in Processing een ovaal te tekenen, gebruik je de functie `ellipse`. De functie `ellipse` heeft vier getallen nodig. De eerste twee getallen zijn de coördinaat van het midden van de ovaal. Het derde getal is de breedte van de ovaal. Het vierde getal is de hoogte van de ovaal.

Hier zie je een ovaal met middelpunt (3,2), een breedte van vijf pixels en hoogte van drie pixels:

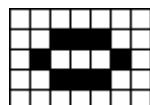


Figure 40: Ovaal 1

In Processing teken je deze ovaal met:

```
ellipse(3,2,5,3);
```

Hier is nog een ovaal:

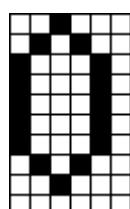


Figure 41: Ovaal 2

Het middelpunt heeft coördinaat (2,4), hij is vijf pixels breed en negen pixels hoog.

Vragen

- 1. Je wilt bovenstaande plaatje namaken. Het venster is 100 pixels breed en 100 pixels hoog. Wat is het middelpunt van de cirkel? Hoe breed is de cirkel? En hoe hoog? Hoe maak je dit in Processing?

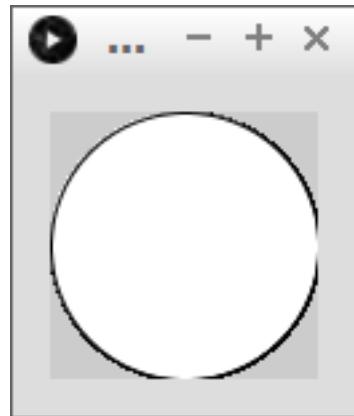


Figure 42: Ovaal 3



Figure 43: Ovaal 4

- 2. Je wilt bovenstaande plaatje namaken. Het venster is 200 pixels breed en 100 pixels hoog. Wat is het middelpunt van de cirkel? Hoe breed is de cirkel? En hoe hoog? Hoe maak je dit in Processing?

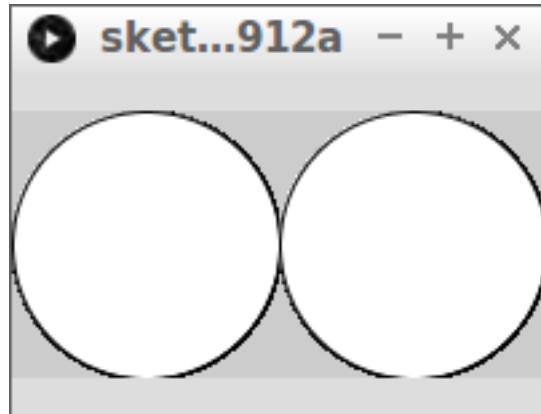


Figure 44: Ovaal 5

- 3. Je wilt bovenstaande plaatje namaken. Het venster is 200 pixels breed en 100 pixels hoog. Wat zijn de middelpunten van de cirkels? Hoe breed zijn de cirkels? En hoe hoog? Hoe maak je dit in Processing?

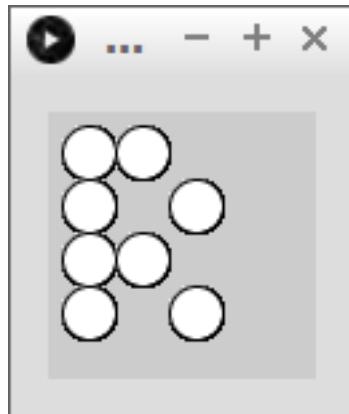


Figure 45: Ellipse

- 4. Hierboven staat een tekening. Maak deze tekening na in Processing

Oplossing

- 1. Zie hieronder:

```

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

void draw() {
    ellipse(50, 50, 100, 100);
}

• 2. Zie hieronder:

void setup() {
    size(200, 100);
}

void draw() {
    ellipse(100, 50, 200, 100);
}

• 3. Zie hieronder:

void setup() {
    size(200, 100);
}

void draw() {
    ellipse( 50, 50, 100, 100);
    ellipse(150, 50, 100, 100);
}

• 4. Zie hieronder:

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

void draw()
{
    ellipse(15,15, 20, 20);
    ellipse(15,35, 20, 20);
    ellipse(15,55, 20, 20);
    ellipse(15,75, 20, 20);

    ellipse(35, 15, 20, 20);
    ellipse(55, 35, 20, 20);
    ellipse(35, 55, 20, 20);
    ellipse(55, 75, 20, 20);
}

```

fill

Een rechthoek of een ovaal kan ook ingekleurd worden.

Een van de beroemdste games ooit heeft bijvoorbeeld veel ingekleurde vierkanten:



Figure 46: Tetris

In deze les gaan we leren

- hoe we de invulkleur van vierkanten en ovalen instellen

Zo gaat het eruit zien:

Weet je nog niet hoe je een lijn een kleur kan geven? Ga dan naar stroke

Kleuren

De lampjes hebben de kleuren rood, groen en blauw.

Omdat de lampjes zo klein zijn, ziet ons oog van afstand de menging van deze drie lampjes. Zo zien de lampjes rood en groen er samen uit als geel. Hier zie je

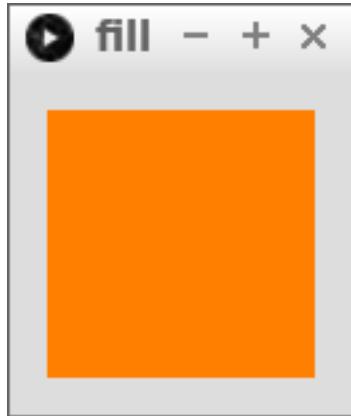


Figure 47: Fill

hoe de kleuren mengen:

Om wit te krijgen, heb je alledrie de kleuren nodig.

Vragen

- 1. Welke drie kleuren lampjes heeft een pixel?
- 2. Met welke lampjes samen maak je geel?
- 3. Met welke lampjes samen maak je cyaan/lichtblauw?
- 4. Met welke lampjes samen maak je magenta/paars?
- 5. Met welke lampjes samen maak je wit?
- 6. Met welke lampjes samen maak je zwart?
- 7. Met welke lampjes samen maak je grijs?
- 8. Met welke lampjes samen maak je oranje?

Antwoorden

- 1. Rood, groen en blauw
- 2. Rood en groen
- 3. Groen en blauw
- 4. Rood en blauw
- 5. Rood en groen en blauw

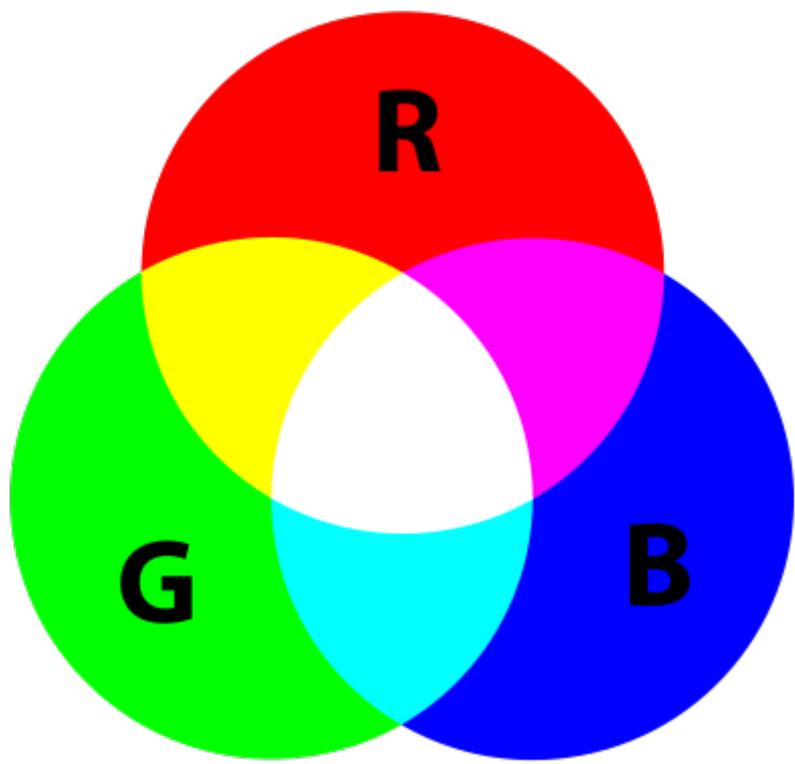


Figure 48: Additieve kleuren

- 6. Met geen lampjes: als alle lampjes uit zijn, is het zwart
- 7. Met rood en groen en blauw, maar dan moeten ze niet op hun hardst branden
- 8. Met rood op z'n hardst en groen op halve kracht

background

In Processing is er een functie om de achtergrond een kleur te geven. Deze functie heet **background**. **background** is Engels voor ‘achtergrond’. **background** is een functie die drie getallen nodig heeft. Deze drie getallen bepalen hoe hard de rode, groene en blauwe lampjes gaan schijnen. Deze drie getallen noemen we de RGB waarde. Met het getal nul zeg je dat een lampje uit staat. Met het getal 255 zeg je dat een lampje aan staat. Met getallen tussen nul en 255 kun je het lampje ertussenin laten branden.

Met deze Processing code krijg je een rode achtergrond:

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

void draw()
{
    background(255,0,0);
}
```

Opdracht

- 1. Kopieer deze code in Processing en start de code
- 2. Wat is een RGB waarde?
- 3. Wat is de RGB waarde van groen? Maak in Processing een groene achtergrond
- 4. Wat is de RGB waarde van blauw? Maak in Processing een blauwe achtergrond
- 5. Wat is de RGB waarde van geel? Maak in Processing een gele achtergrond
- 6. Wat is de RGB waarde van cyaan/lichtblauw? Maak in Processing een cyane/lichtblauwe achtergrond
- 7. Wat is de RGB waarde van magenta/paars? Maak in Processing een magenta/paarse achtergrond

- 8. Wat is de RGB waarde van wit? Maak in Processing een witte achtergrond
- 9. Wat is de RGB waarde van zwart? Maak in Processing een zwarte achtergrond
- 10. Wat is de RGB waarde van grijs? Maak in Processing een grijze achtergrond
- 11. Wat is de RGB waarde van donkerrood? Maak in Processing een donkerrode achtergrond
- 12. Wat is de RGB waarde van oranje? Maak in Processing een oranje achtergrond

Oplossingen

- 1. OK
- 2. De Rood-Groen-Blauw waarde. Dit zijn drie getallen van nul tot en met 255 die bepalen hoe hard de drie kleurenlampjes branden
- 3. background(0,255,0)
- 4. background(0,0,255)
- 5. background(255,255,0)
- 6. background(0,255,255)
- 7. background(255,255,0)
- 8. background(255,255,255)
- 9. background(0,0,0)
- 10. background(128,128,128), maar andere getallen tussen 0 en 255 zijn ook goed. Als ze maar alledrie gelijk zijn
- 11. background(128,0,0), maar het eerste getal mag ook een ander getal tussen de 0 en 255 zijn
- 12. background(255,128,0), maar het tweede getal mag ook een ander getal in de buurt van 128 zijn

Bal naar rechts

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

Het ziet er zo uit:

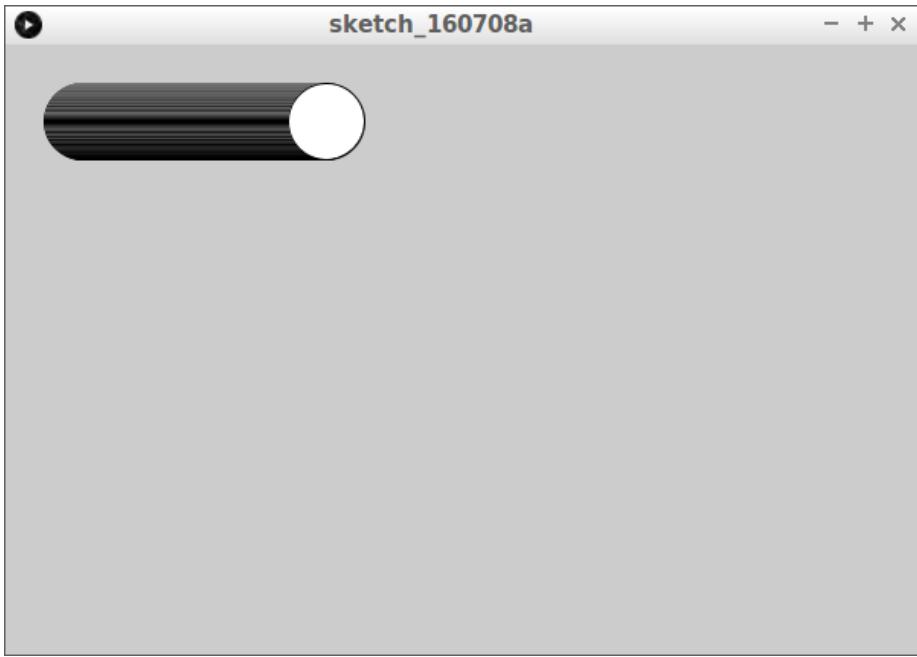


Figure 49: Bal naar rechts

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

- Wat doet dit programma?

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

We gaan de code aanpassen:

```
float x = 50;

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

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

Vragen

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

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

- Wat als je `float x = 50;` weghaalt?
- Wat als je `float x = 50;` verandert naar `float rechts = 50;?`
- Wat als je `float x = 50;` verandert naar `float x = 100;?`
- Wat als je `ellipse(x,50,50,50);` weghaalt?
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(rechts,50,50,50);?`
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,50,50);?`
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,50);?`
- Wat als je `ellipse(x,50,50,50);` verandert naar `ellipse(x,x,x,x);?`
- Wat als je `x` vervangt door `rechts`?
- Wat als je `x` vervangt door `dinosaurus`?

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,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 $x = x / 0$;?

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 50: Bal die eeuwig naar rechts gaat 1

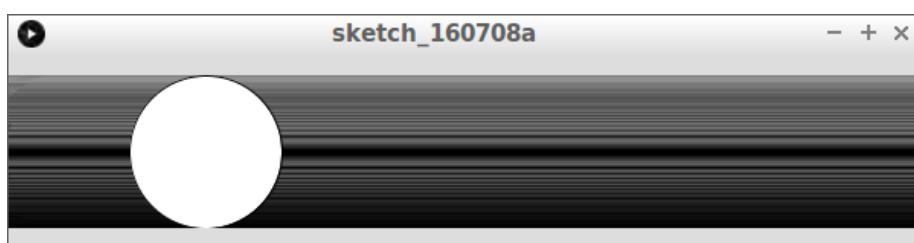


Figure 51: Bal die eeuwig naar rechts gaat 2

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 rechts 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 :-)

Verder

Je zou nu kunnen doen:

- Bal die horizontaal stuiter

Bal die horizontaal stuiter

In deze les gaan we een bal horizontaal laten stuiteren.

Het ziet er zo uit:

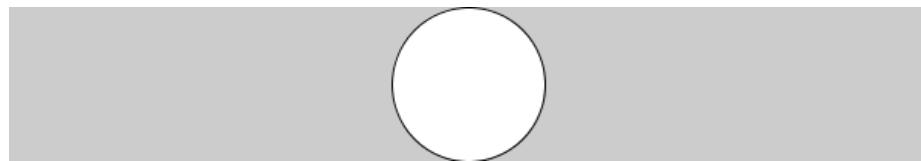


Figure 52: Bal die horizontaal stuiter (zie ‘dojo/Images/BalDieHorizontaalStuitertGif’)

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

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 rechts gaat en stuert, 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. Stuert 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).

Opdracht

- 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:

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);
```

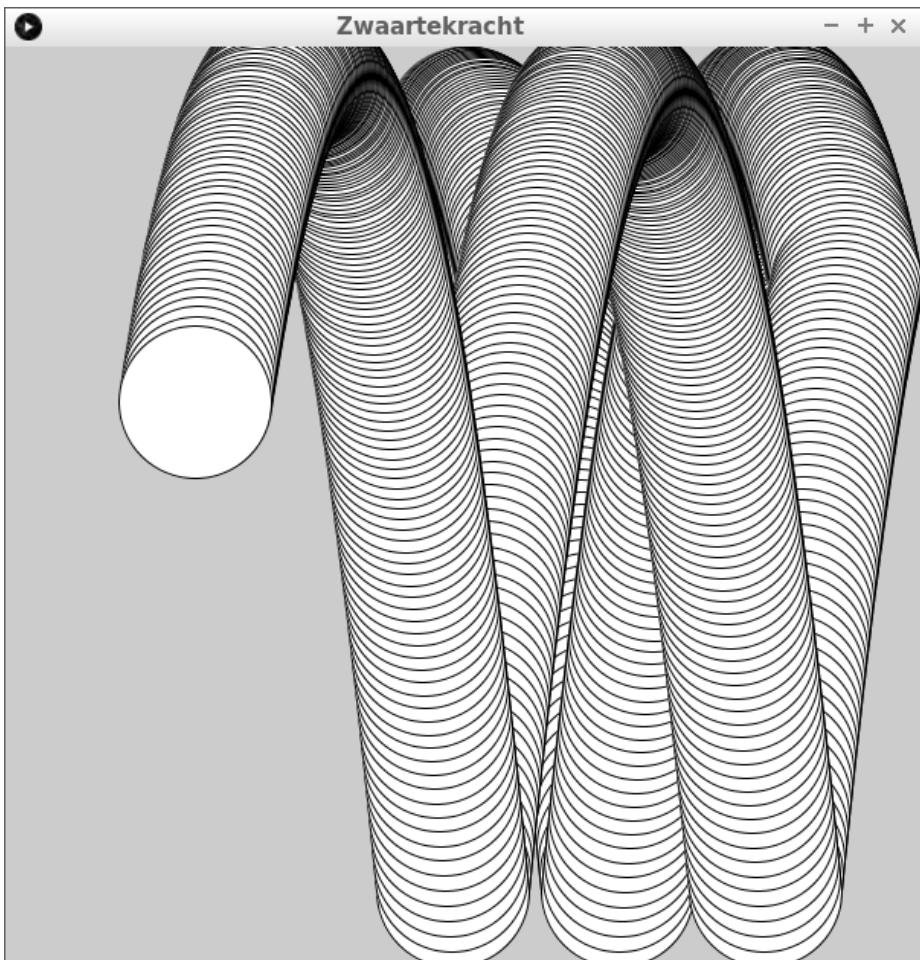


Figure 53: Zwaartekracht

```

}

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

```

Vragen

- Wat doet dit programma?
- In welke richting beweegt de ovaal?
- Blijft de ovaal zichtbaar op het scherm?

Opdrachten

- Laat de bal nu *ook* verticaal bewegen. Gebruik de variabele namen y en dy. Zorg dat de bal ook verticaal stuert

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 natuurkunde gebruiken ze g (van ‘gravity’, dit is Engels voor zwaartekracht) voor de zwaartekracht.

Opdacht

In onze code hebben we nu een dy. Dit is de snelheid in de y richting. Elke beurt moet dy nu g groter worden. Een mooie waarde voor g is 0.1.

Voeg toe dat de cirkel op een natuurlijke manier omlaag valt.

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 rechts gaat en stuert, 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).

Opdracht

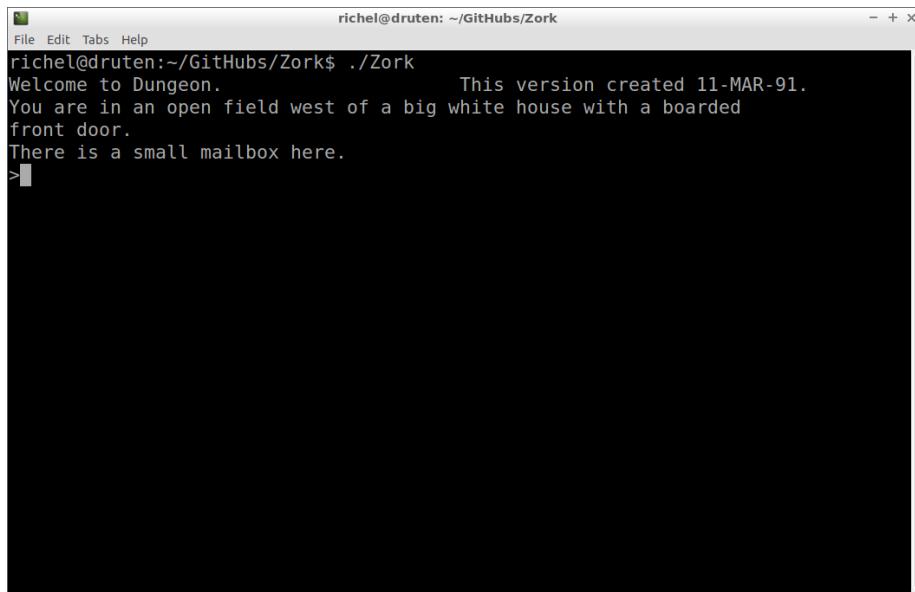
- Gebruik nu de slimme manier om een bal te laten stuiteren.

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

text

Tekst wordt veel gebruikt, ook in games, voor bijvoorbeeld een score.

Hier zie je 'Zork, the underground empire', een van de beroemdste tekstavonturen ooit:



A screenshot of a terminal window titled 'richel@druten: ~/GitHub/Zork'. The window contains the following text:
richel@druten:~/GitHub/Zork\$./Zork
Welcome to Dungeon. This version created 11-MAR-91.
You are in an open field west of a big white house with a boarded
front door.
There is a small mailbox here.
>|

Figure 54: Zork

In deze les gaan we leren

- hoe je tekst op het scherm zet
- hoe je berekeningen op het scherm zet
- hoe je tekst vergroot
- hoe je tekst een kleur geeft

Zo gaat het eruit zien:

Kun je nog geen puntjes tekenen? Ga dan naar de les waarin je puntjes leert tekenen

Kun je nog geen vlakken inkleuren? Ga dan naar de les 'fill'

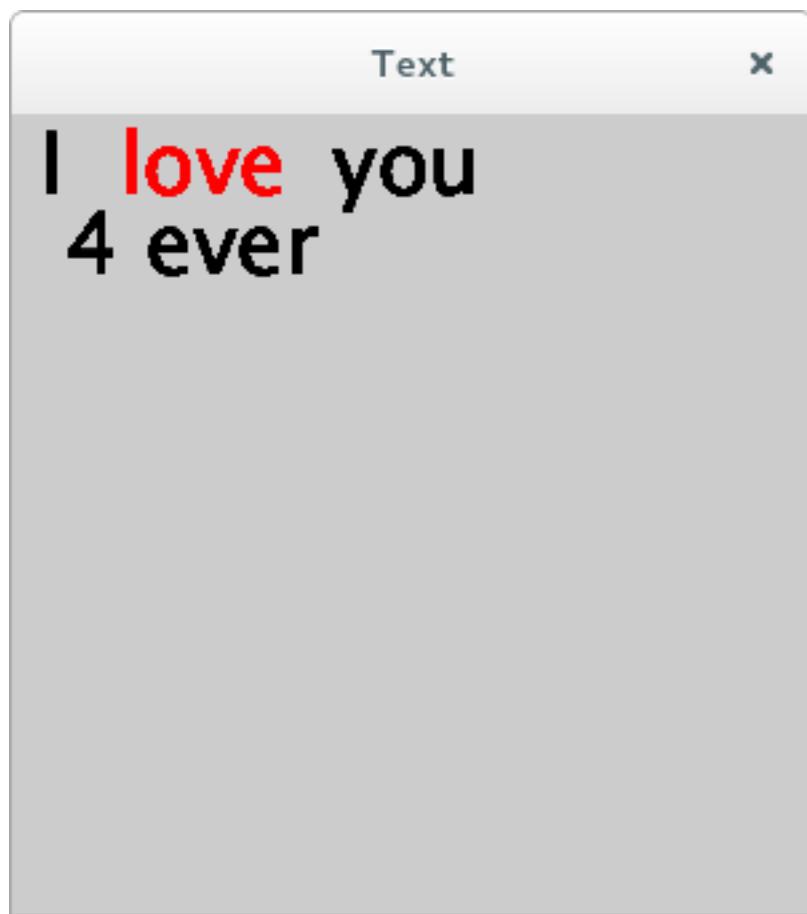


Figure 55: Text

Tekst

Hier zie je de tekst ‘Hallo’ laat zetten op coordinaat (10,20):

```
text("Hallo", 10, 20);
```

Let op dat de tekst tussen dubbele apostroffen ("") moet.

text kan ook rekenen!

Hier plus en min:

```
text(128 + 64, 10, 20);  
text(128 - 64, 10, 20);
```

Hier een keersom:

```
text(16 * 16, 10, 20);
```

Hier een deelsom:

```
text(256 / 16, 10, 20);
```

Tekstgrootte kun je aanpassen met

```
textSize(32);
```

Tekstkleur kun je aanpassen met **fill**:

```
fill(255, 0, 0);
```

Opdracht

Zet de tekst **I love you 4 ever** op het scherm, waarbij:

- alle woorden zwart zijn, behalve **love**, die rood is
- de **4** is de uitkomst van een berekening, bijvoorbeeld **2 + 2** (maar hoe moeilijker de berekening, hoe stoerder)

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:

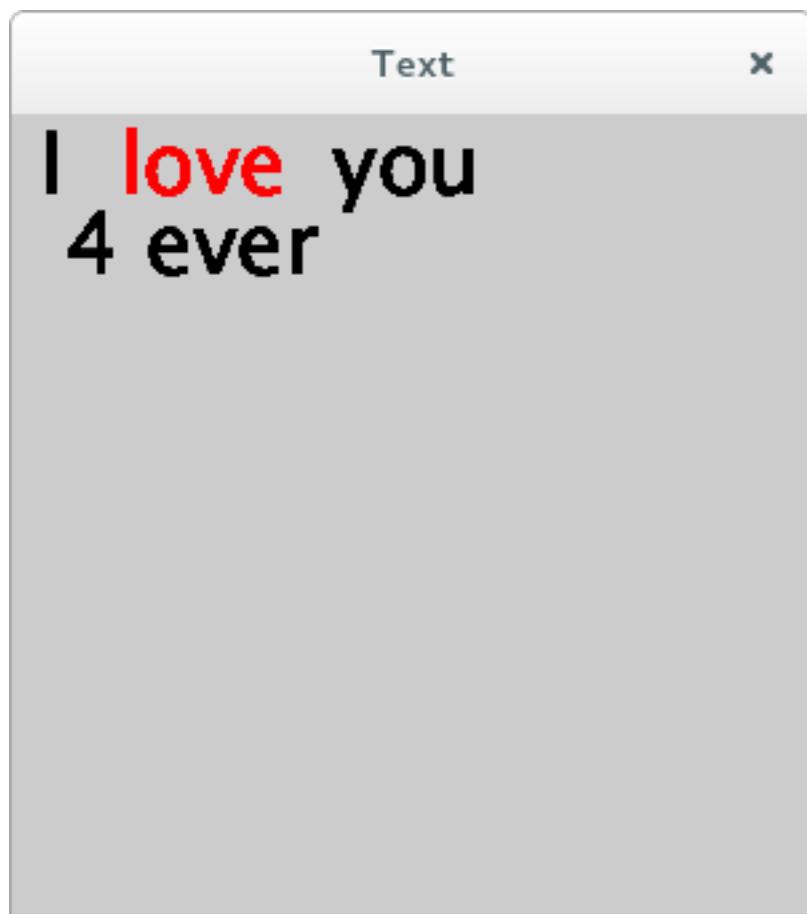


Figure 56: Text

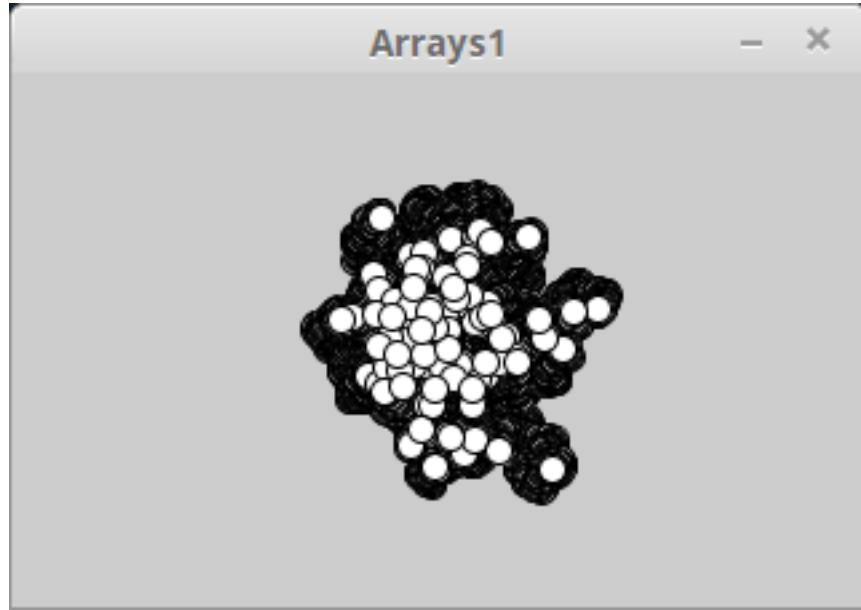


Figure 57: Arrays1

Rooksimulator met een deeltje

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

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 accolades 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 accolades 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

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

Rooksimulator met twee deeltjes

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

Dit is je code:

```
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);
}

```

Dit is wat de code betekent:

- **float x1 = 160:** ‘Lieve computer, onthoudt een gebroken getal met de naam **x1**, met als beginwaarde 160’. Dit wordt de x coordinaat van het eerste rookdeeltje
- **float y1 = 100:** ‘Lieve computer, onthoudt een gebroken getal met de naam **y1**, met als beginwaarde 100’. Dit wordt de y coordinaat van het eerste rookdeeltje
- **float x2 = 160:** ‘Lieve computer, onthoudt een gebroken getal met de naam **x2**, met als beginwaarde 160’. Dit wordt de x coordinaat van het tweede rookdeeltje
- **float y2 = 100:** ‘Lieve computer, onthoudt een gebroken getal met de naam **y2**, met als beginwaarde 100’. Dit wordt de y coordinaat van het tweede 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
- **x1 += random(-1,1):** verander de waarde van **x1** met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig horizontaal bewegen
- **y1 += random(-1,1):** verander de waarde van **y1** met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig verticaal bewegen
- **ellipse(x1, y1, 10, 10):** teken een ovaal met als middelpunt (**x1, y1**) met breedte 10 en hoogte 10. Teken het eerste rookdeeltje
- **x2 += random(-1,1):** verander de waarde van **x2** met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 2 willekeurig horizontaal bewegen
- **y2 += random(-1,1):** verander de waarde van **y2** met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 2 willekeurig verticaal bewegen
- **ellipse(x2, y2, 10, 10):** teken een ovaal met als middelpunt (**x2, y2**) met breedte 10 en hoogte 10. Teken het tweede rookdeeltje

Vragen

- Run deze code
- Zorg dat er een derde rookdeeltje bijkomt
- Hoeveel regels code kost het ongeveer om honderd rookdeeltjes te programmeren?

Waarom arrays?

Dit is de code met drie rookdeeltjes:

```
float x1 = 160;
float y1 = 100;
float x2 = 160;
float y2 = 100;
float x3 = 160;
float y3 = 100;

void setup()
{
    size(320, 200);
    x1 = 160;
    y1 = 100;
    x2 = 160;
    y2 = 100;
    x3 = 160;
    y3 = 100;
}

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);
    x3 += random(-1,1);
    y3 += random(-1,1);
    ellipse(x3, y3, 10, 10);
}
```

Het valt op dat er veel herhaling in zit. Dit komt omdat we de computer steeds een getal tegelijk laten onthouden: `float x1 = 160` betekent ‘Lieve computer, onthoudt een gebroken getal met de naam `x1`, met als beginwaarde 160’. 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 58: 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:

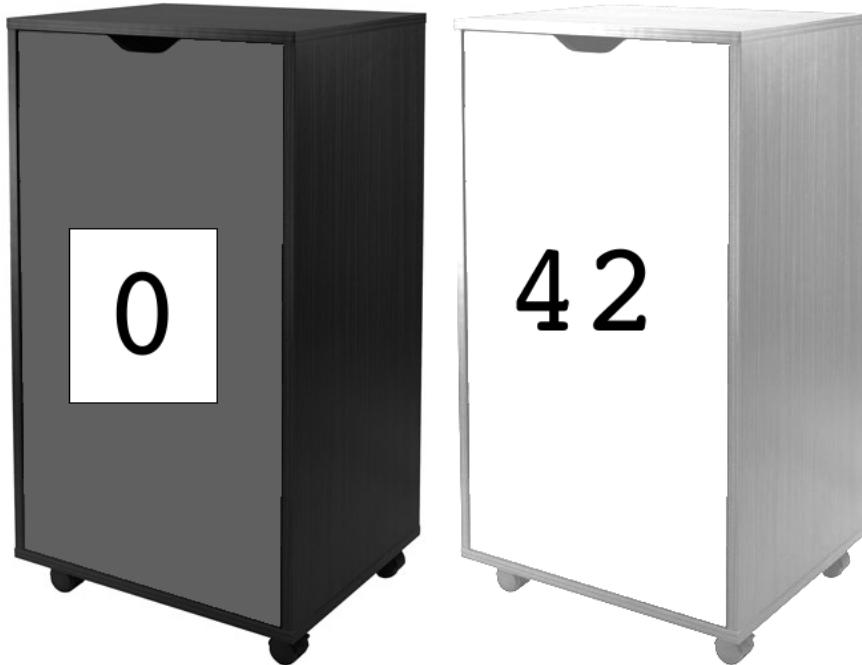


Figure 59: Kast met genummerde laatjes

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 één. 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;
```

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;
}

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?

Verder

Je zou nu kunnen doen:

- Arrays 2

Arrays2

Met arrays kun je de computer veel waardes laten onthouden: de coördinaten van kogels, meteorieten, vijanden. Met een for loop kun je door een array heen gaan.

In deze les gaan we leren

- waarom je arrays nodig hebt
- wat arrays zijn
- hoe je arrays met een for loop gebruikt

Zo gaat het eruit zien:

Rooksimulator

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

Dit is je code:

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

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

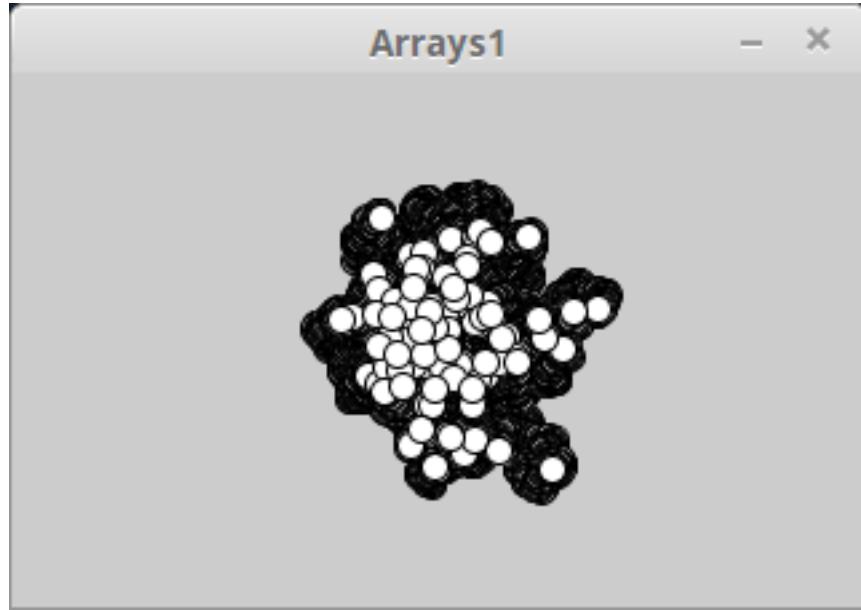


Figure 60: Arrays2

```
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);
}
```

Dit is wat de code betekent:

- **float x1 = 160;** ‘Lieve computer, onthoudt een gebroken getal met de naam **x1**, met als beginwaarde 160’. Dit wordt de x coordinaat van het eerste rookdeeltje
- **float y1 = 100;** ‘Lieve computer, onthoudt een gebroken getal met de naam **y1**, met als beginwaarde 100’. Dit wordt de y coordinaat van het eerste rookdeeltje
- **float x2 = 160;** ‘Lieve computer, onthoudt een gebroken getal met de naam **x2**, met als beginwaarde 160’. Dit wordt de x coordinaat van het tweede rookdeeltje
- **float y2 = 100;** ‘Lieve computer, onthoudt een gebroken getal met de naam **y2**, met als beginwaarde 100’. Dit wordt de y coordinaat van het

tweede rookdeeltje

- `void setup() {}`: de klaarzet functie. Bij het opstarten wordt de code tussen de accolades 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 accolades wordt oneindig vaak uitgevoerd
- `x1 += random(-1,1)`: verander de waarde van `x1` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig horizontaal bewegen
- `y1 += random(-1,1)`: verander de waarde van `y1` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 1 willekeurig verticaal bewegen
- `ellipse(x1, y1, 10, 10)`: teken een ovaal met als middelpunt (`x1, y1`) met breedte 10 en hoogte 10. Teken het eerste rookdeeltje
- `x2 += random(-1,1)`: verander de waarde van `x2` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 2 willekeurig horizontaal bewegen
- `y2 += random(-1,1)`: verander de waarde van `y2` met een willekeurige waarden van -1 tot 1. Dit laat rookdeeltje 2 willekeurig verticaal bewegen
- `ellipse(x2, y2, 10, 10)`: teken een ovaal met als middelpunt (`x2, y2`) met breedte 10 en hoogte 10. Teken het tweede rookdeeltje

Vragen

- Run deze code
- Zorg dat er een derde rookdeeltje bijkomt
- Hoeveel regels code kost het ongeveer om honderd rookdeeltjes te programmeren?

Waarom arrays?

Dit is de code met drie rookdeeltjes:

```
float x1 = 160;
float y1 = 100;
float x2 = 160;
float y2 = 100;
float x3 = 160;
float y3 = 100;

void setup()
{
    size(320, 200);
    x1 = 160;
    y1 = 100;
    x2 = 160;
```

```

y2 = 100;
x3 = 160;
y3 = 100;
}

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);
    x3 += random(-1,1);
    y3 += random(-1,1);
    ellipse(x3, y3, 10, 10);
}

```

Het valt op dat er veel herhaling in zit. Dit komt omdat we de computer steeds een getal tegelijk laten onthouden: `float x1 = 160` betekent ‘Lieve computer, onthoud een gebroken getal met de naam `x1`, met als beginwaarde 160’. 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:

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

Hier zie je de nummers van de laatjes:

Het valt op dat het eerste 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 zeven laatjes, met indices nul tot en met zes.

Vragen

- Wat is een array?
- Wat is een index?
- Een array kan drie getallen bevatten. Wat zijn de indices?
- Een array heeft als hoogste index 13. Wat is de grootte van de array?
- Een array kan 314 getallen bevatten. Wat is de hoogste index?
- Een array heeft als hoogste index 0. Wat is de grootte van de array?



Figure 61: Kast met laatjes



Figure 62: Kast met genummerde laatjes

Werken met arrays

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[7];
```

Hiermee zeg je: ‘Lieve computer, maak **geheime_getallen** groot genoeg om zeven gebroken getallen (**floats**) te onthouden’.

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

```
geheime_getallen[0] = 0;
geheime_getallen[1] = 1;
geheime_getallen[2] = 4;
geheime_getallen[3] = 9;
geheime_getallen[4] = 16;
geheime_getallen[5] = 25;
geheime_getallen[6] = 36;
```

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

Dit kan ook slimmer. Hieronder zie je code die *precies* hetzelfde doet:

```
for (int i=0; i!=7; ++i)
{
    geheime_getallen[i] = i * i;
```

Hiermee zeg je: ‘Lieve computer, laat een for loop lopen van nul tot zeven met **i** als teller. Stop in het **i**-de laatje het getal **i** keer **i**’.

Je zou ook de waarden in de laatjes kunnen lezen:

```
for (int i=0; i!=7; ++i)
{
    float x = geheime_getallen[i];
    ellipse(x,200,300,400);
}
```

Hiermee zeg je: ‘Lieve computer, laat een for loop lopen van nul tot zeven met **i** als teller. Kijk wat er in het **i**-de laatje zit en onthoud dat als **x**. Tekent dan een

oval 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[7];
    for (int i=0; i!=7; ++i)
    {
        geheime_getallen[i] = i * i;
    }
}

void draw()
{
    for (int i=0; i!=7; ++i)
    {
        float x = geheime_getallen[i];
        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?
- Je hebt een array van gebroken getallen met de naam `schades`. Je wilt dat deze 345 groot wordt. Hoe zeg je dat in code?
- Je hebt een array van gebroken getallen met de naam `roodwaarden` die 987 groot is. Je wilt alle `roodwaarden` op 128 zetten. Hoe doe je dat?
- Je hebt een array van gebroken getallen met de naam `breedtes` die 345 groot is. Je wilt 345 ovalen tekenen, op 123 pixels naar rechts, 234 pixels omlaag, met een breedte van `breedtes` en een hoogte van 345. Hoe doe je dat?

Code met drie rookdeeltjes

Kijk naar de code met de drie rookdeeltjes:

```
float x1 = 160;
float y1 = 100;
float x2 = 160;
float y2 = 100;
float x3 = 160;
float y3 = 100;

void setup()
{
    size(320, 200);
    x1 = 160;
    y1 = 100;
    x2 = 160;
    y2 = 100;
    x3 = 160;
    y3 = 100;
}

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);
    x3 += random(-1,1);
    y3 += random(-1,1);
    ellipse(x3, y3, 10, 10);
}
```

Opdracht

- Schrijf deze code om dat deze gebruik maakt van arrays
- Verander de code zo dat er honderd rookdeeltjes kunnen zijn

Oplossing

Hier de oplossing:

```
final int aantal_punten = 100;
```

```

float[] xs;
float[] ys;

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

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

```

Dit doet de code:

- **final int aantal_punten = 100;** 'Lieve computer, onthoud een heel getal (**int**), die ik **aantal_punten** noem, met beginwaarde 100, die ik nooit zal veranderen (**final**)'
- **float[] xs;** 'Lieve computer, onthoud een heleboel gebroken getallen (**float[]**), die ik **xs** noem'. Dit zijn de x-coordinaten (hoeveel pixels naar rechts) van de rookdeeltjes.
- **float[] ys;** 'Lieve computer, onthoud een heleboel gebroken getallen (**float[]**), die ik **ys** noem'. Dit zijn de y-coordinaten (hoeveel pixels omlaag) van de rookdeeltjes.
- **size(320, 200);** maak het scherm 320 pixels breed en 200 pixels hoog
- **xs = new float[aantal_punten];** 'Lieve computer, maak **xs** groot genoeg om **aantal_punten** (dus 100) gebroken getallen (**floats**) te onthouden'
- **ys = new float[aantal_punten];** 'Lieve computer, maak **ys** groot genoeg om **aantal_punten** (dus 100) gebroken getallen (**floats**) te onthouden'

```

for (int i=0; i!=aantal_punten; ++i)
{
    xs[i] = 160;
}

```

```
    ys[i] = 100;  
}
```

- Laat een teller lopen van nul tot aantal_punten (dus 100) in stappen van een en noem het tellertje i. Zet de i-de xs op 160. Zet de i-de ys op 100.

```
for (int i=0; i!=aantal_punten; ++i)  
{  
    xs[i] += random(-1,1);  
    ys[i] += random(-1,1);  
    ellipse(xs[i], ys[i], 10, 10);  
}
```

- Laat een teller lopen van nul tot aantal_punten (dus 100) in stappen van een en noem het tellertje i. Verander de i-de xs willekeurig met -1, 0 of +1. Verander de i-de ys willekeurig met -1, 0 of +1. Teken een ellise met de i-de xs naar rechts, de i-de ys omlaag, 10 pixels breed en 10 pixels hoog

PIImage les 1

In deze les gaan we met een plaatje werken.

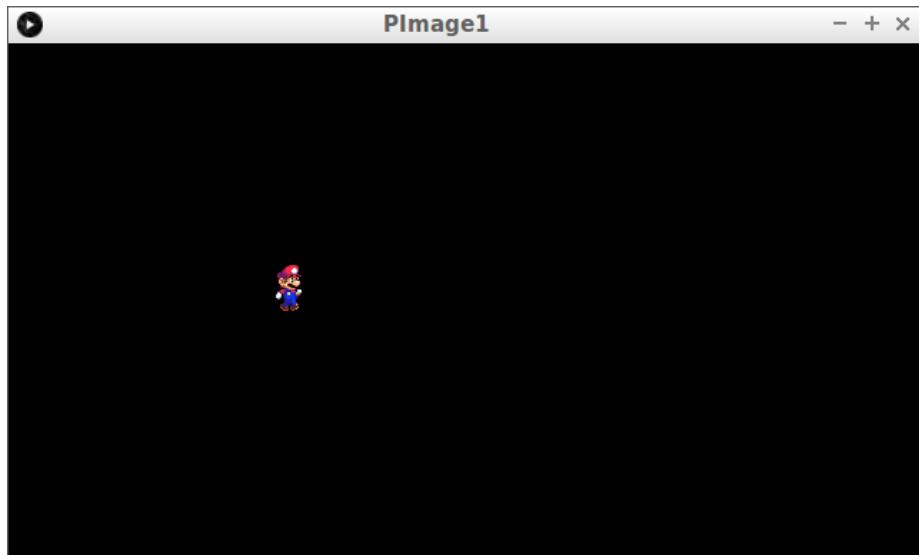


Figure 63: PImage1.png

Een plaatje vinden

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

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 64: 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(plateje, mouseX, mouseY);  
}
```

- `PImage plaatje`: onthoud een `PImage` met de naam `plateje`. 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
- `image(plateje, mouseX, mouseY)`: Laat `plateje` 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:

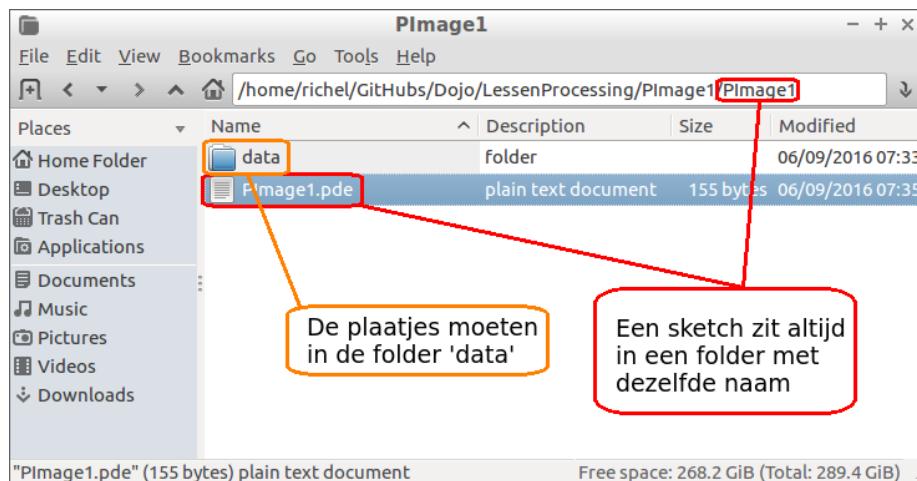


Figure 65: Folderstructuur.png

- De sketch heet `PImage1.pde`. Daarom staat deze in de map `PImage1`
- 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

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:

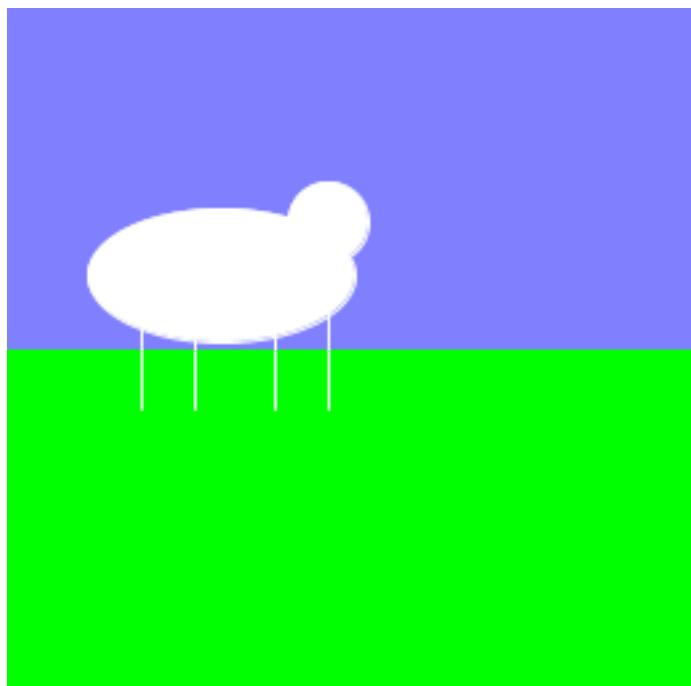


Figure 66: Wit Schaap

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  
    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 hypothese 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

- Kopieér 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

GitHub

GitHub is een site waar programmeurs samen aan programma's werken. Dit doen ze, omdat ze dan nooit meer hun code kwijtraken.

Wij gaan vandaag GitHub gebruiken om:

- Nooit meer code kwijt te raken
- Altijd bij onze code te kunnen

Wij gaan later GitHub gebruiken om:

- Samen te kunnen werken
- Onze games online te kunnen zetten

:exclamation: Waarschuwing :exclamation:

GitHub is openbaar. Iedereen kan zien wat je doet. Doe dus alleen slimme dingen op GitHub:

- Blijf altijd vriendelijk
- Blijf altijd netjes
- Grappen zijn prima, maar nooit als iemand zich beledigd voelt

GitHub aanmaken

In deze les maken we een GitHub account en gaan we onze code er op zetten.

Ga naar GitHub

- Start een webbrowser, zoals Firefox, Chromium, Chrome, Edge, Internet Explorer, Opera
- Ga naar www.github.com

Je komt nu op de GitHub homepage:

De kat heet Octocat en is de mascotte van GitHub.

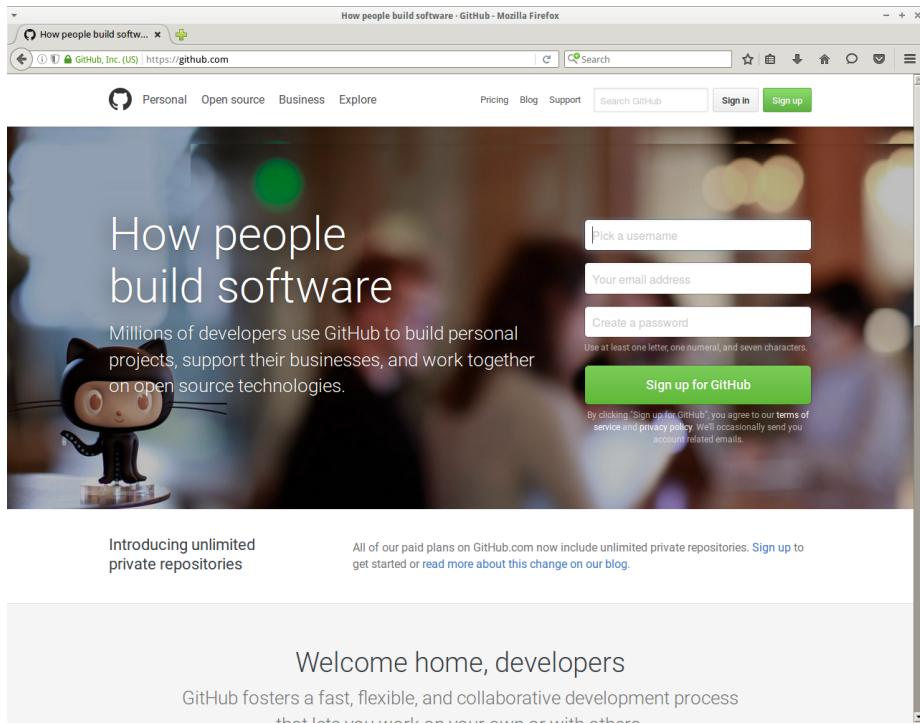


Figure 67: GitHub homepage

Maak jezelf lid

- Klik op Sign Up (NL: Inschrijven)

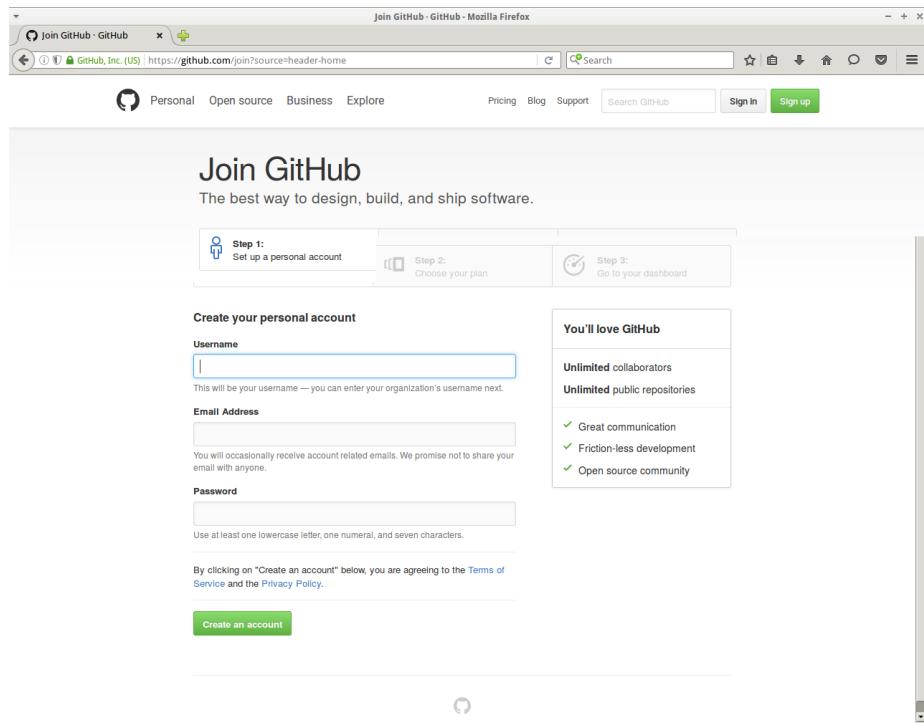


Figure 68: GitHub inschrijven

Vul hier je gegevens in:

- **Username:** je gebruikersnaam. Kies geen gebruikersnaam waar je later spijt van gaat krijgen
- **Email address:** je emailadres. Dit moet je echte emailadres zijn
- **Password:** je wachtwoord. Deze moet een cijfer bevatten, bijvoorbeeld `iloverichel4ever` is een goede
- Klik **Create an account** om een account aan te maken.

Kies je account

In het volgende scherm moet je je soort account kiezen.

Wij gebruiken gewoon een gratis account

- Klik **Finish sign up** en je hebt een GitHub account!

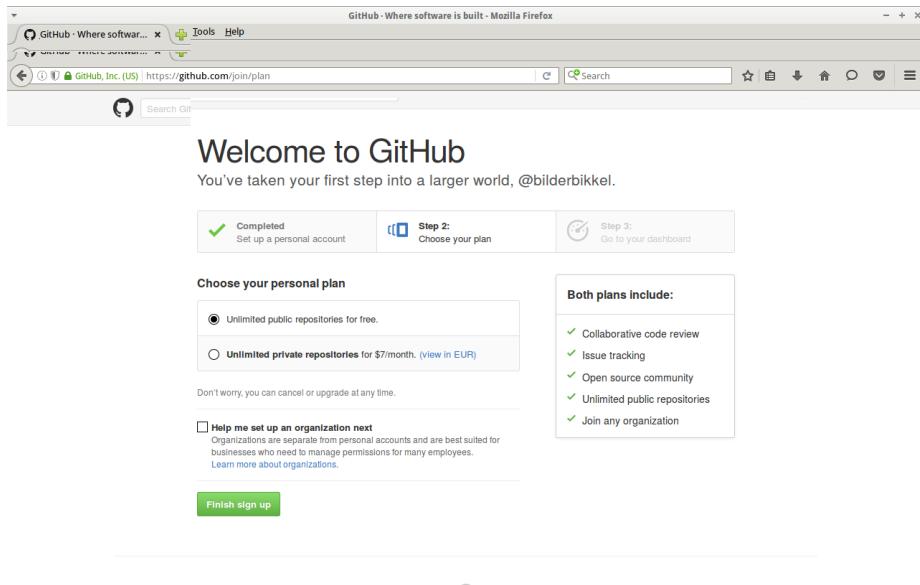


Figure 69: GitHub account kiezen

Vrienden maken

Nu je een account hebt, kun je vrienden maken

- Ga naar <https://github.com/richelbilderbeek> en klik op Follow om Richel te volgen
- Ga naar <https://github.com/thijsvb> en klik op Follow om Thijs te volgen
- Ga naar <https://github.com/LJK1991> en klik op Follow om Lucas te volgen
- Ga naar <https://github.com/Modanung> en klik op Follow om Frode te volgen

Bevestig je emailadres

- Ga naar je email toe
- Open het maitje van GitHub
- Klik op Confirm om te bevestigen dat je een GitHub account hebt

Een eigen GitHub website aanmaken

:exclamation: Dit kan alleen als je je emailadres bevestigd hebt

Maak een nieuwe GitHub website

In bijna elk scherm, is er een kruis rechts bovenaan:

- Klik op Create New Repository om een nieuwe GitHub webpagina aan te maken

Stel de nieuwe GitHub website in

Vul hier in:

- Repository name: de naam van je GitHub website, bijvoorbeeld Mijncraft
- Description: omschrijving van je GitHub website, bijvoorbeeld Minecraft-achtig spel, gemaakt in Processing
- Kies Public
- Vink aan: Initialize this repository with a README
- Kies bij Add .gitignore: Processing
- Kies bij Add a license: GNU General Public License v3.0

Nu heb je je eigen GitHub website!

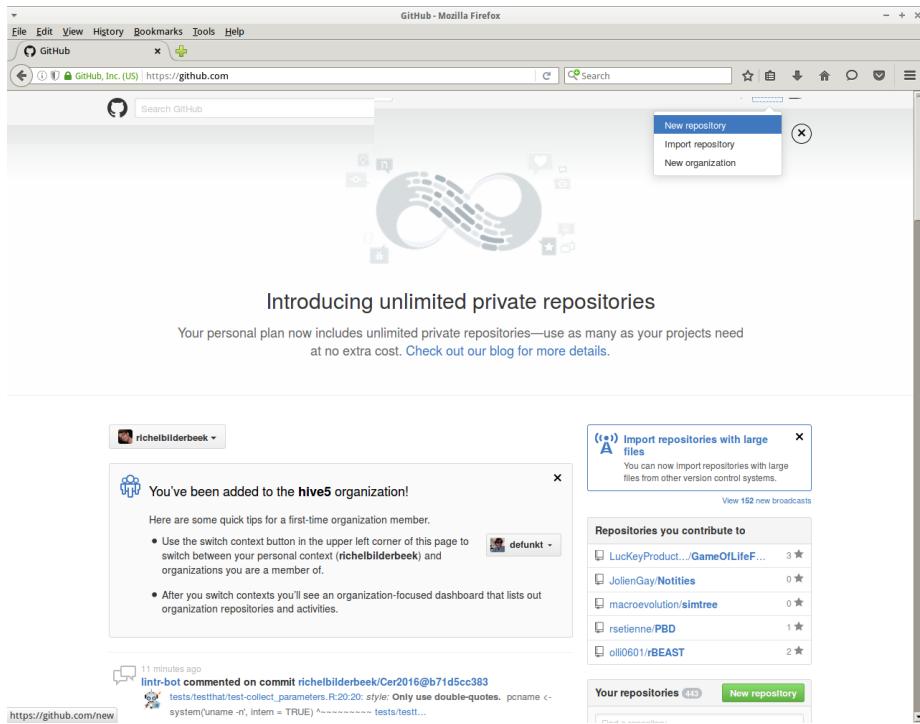


Figure 70: Maak een GitHub

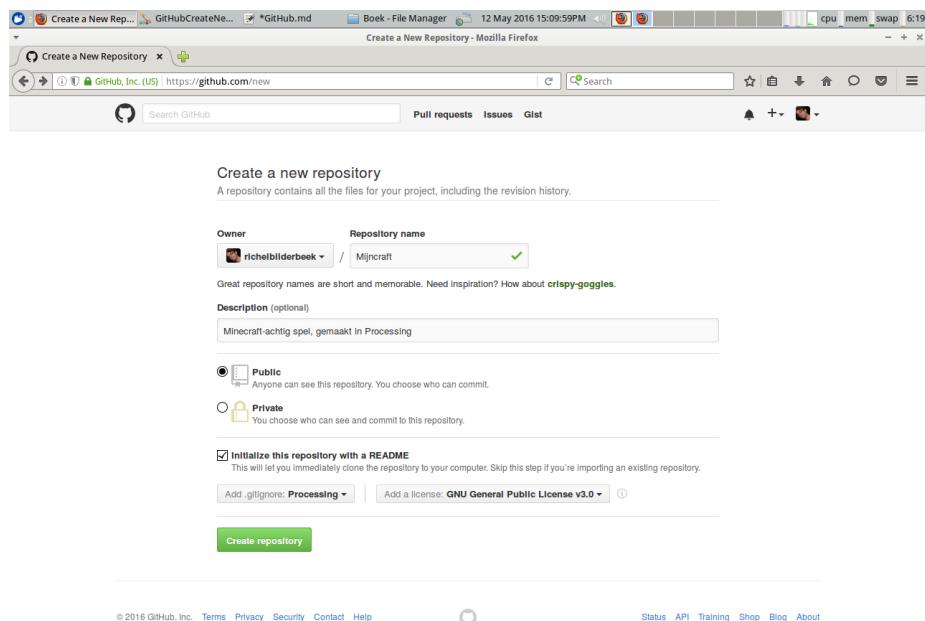


Figure 71: Maak een GitHub, eerste scherm

Zet dingen op je eigen GitHub website

Op je eigen GitHub website kun je veel dingen doen.

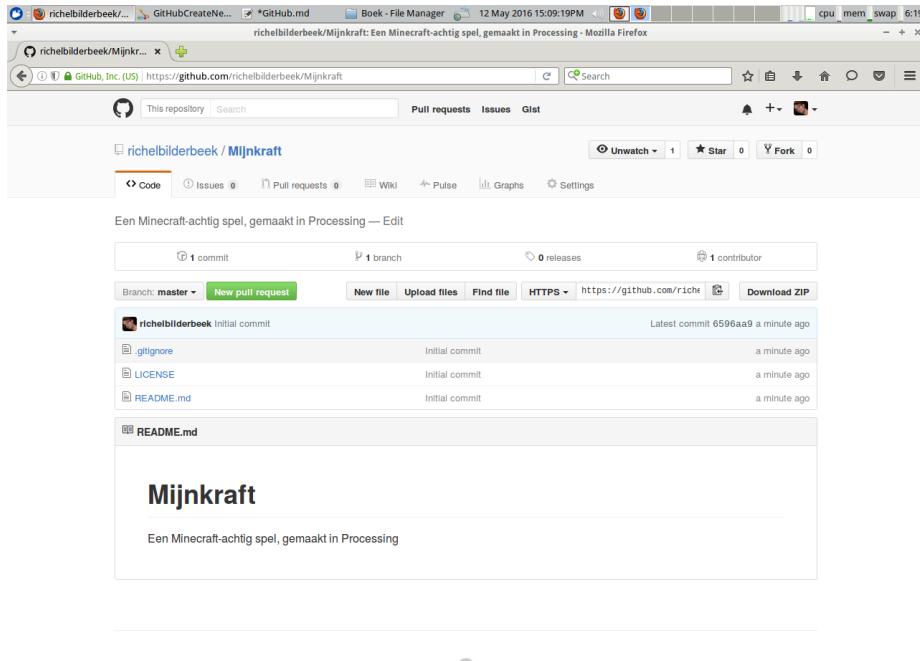


Figure 72: De Mijnkraft GitHub

- Klik op **Upload file** om een bestand op de website te zetten. Zet een mooi plaatje op de website.

Het bestand **README.md** bevat de welkomsttekst van je website. Dit gaan we aanpassen.

- Klik op het woord **README.md**
- Klik op het potloodje met de tekst **Edit this file**
- Zet hier iets leuks. Je kunt ook je plaatje laten zien, met bijvoorbeeld **! [Dit vind ik leuk] (plaatje.jpg)**
- Zet je code op GitHub

Als je nog geen GitHub account hebt, doe dan eerst deze les

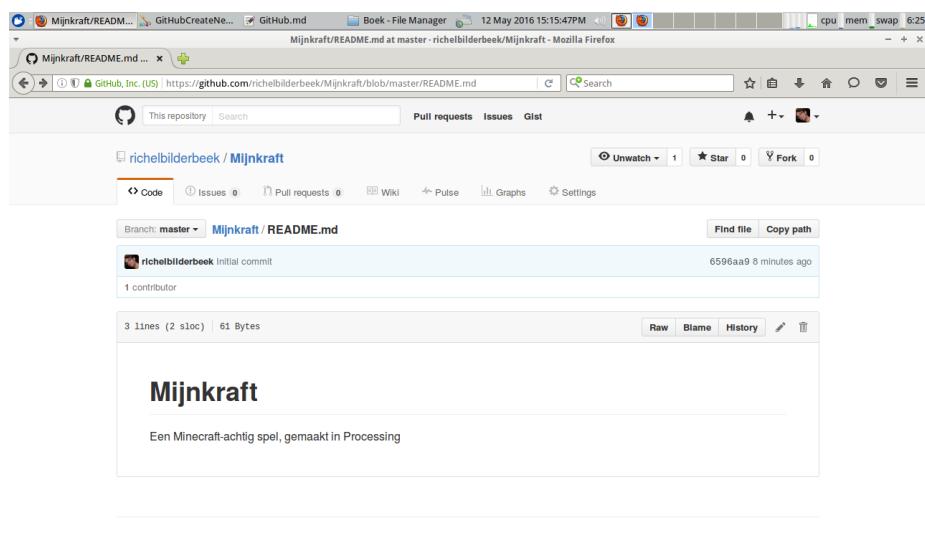


Figure 73: README.md wijzigen

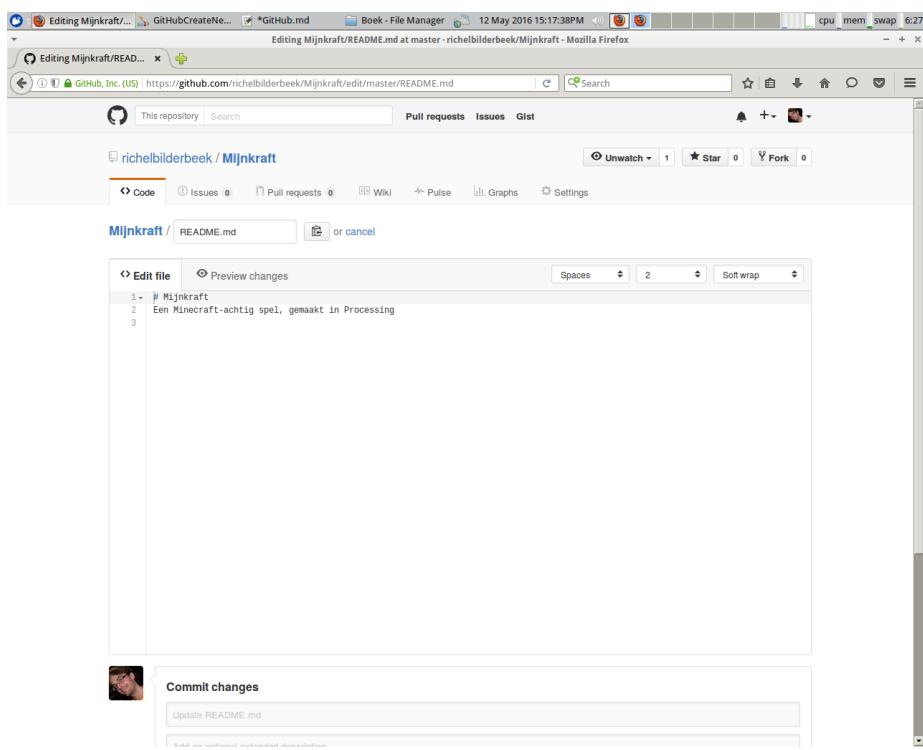


Figure 74: README.md wijzigen

GitHub Pages

GitHub Pages is een service van GitHub waarmee elke gebruiker zijn eigen website kan maken. Deze les gaan we een GitHub Page met homepage maken.

Het verschil met een repository is dat die op www.github.com/gebruiker staat, terwijl een GitHub Page op www.gebruiker.github.io staat.

Een repository gebruik je om:

- * Je code te bewaren
- * Je code te delen met de wereld
- * Samen aan code te werken

Een GitHub Page gebruik je om:

- * Mensen jouw spellen te laten spelen
- * De wereld te laten zien wat je kan

GitHub Page aanmaken

- Open een webbrowser
- Ga naar www.github.com

Je komt nu op de GitHub homepage:

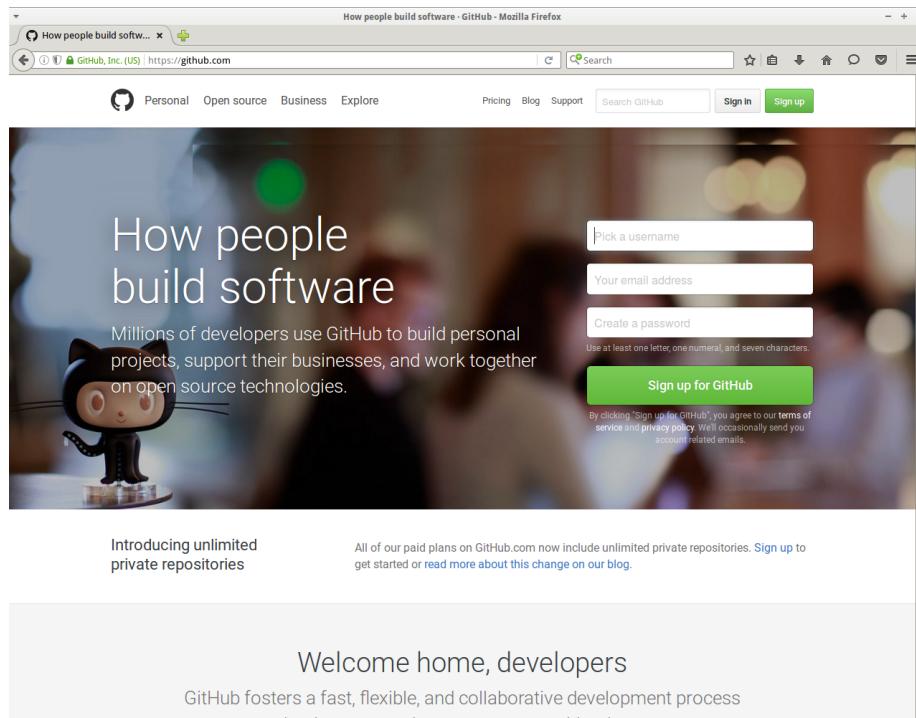


Figure 75: GitHub homepage

- Klik op **Sign in** (NL: Aanmelden)

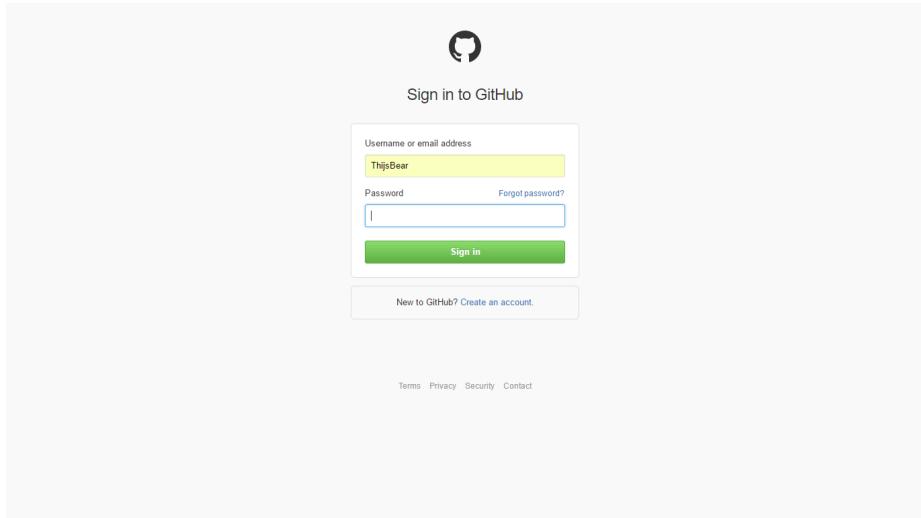


Figure 76: GitHub aanmelden

- Log in met je GitHub account (Heb je die niet? Doe dan eerst deze les!)

GitHub Page repository maken

- Klik op het plusje bovenin
- Klik op **New repository**

Vul hier in:

- **Repository name:** de naam van je GitHub Page moet zijn `gebruiker.github.io`, dus bijvoorbeeld `thijsvb.github.io`
- **Description:** omschrijving van je GitHub Page, bijvoorbeeld `Mijn geweldige website!`
- Kies `Public`
- Vink aan: `Initialize this repository with a README`
- Kies bij `Add a license: GNU General Public License v3.0`

Je hebt nu een GitHub Page!

HTML

Als je nu in een browser naar je GitHub Page gaat, is er nog niks. Je moet een homepage maken, en om dat te doen moet je HTML-bestanden kunnen schrijven.

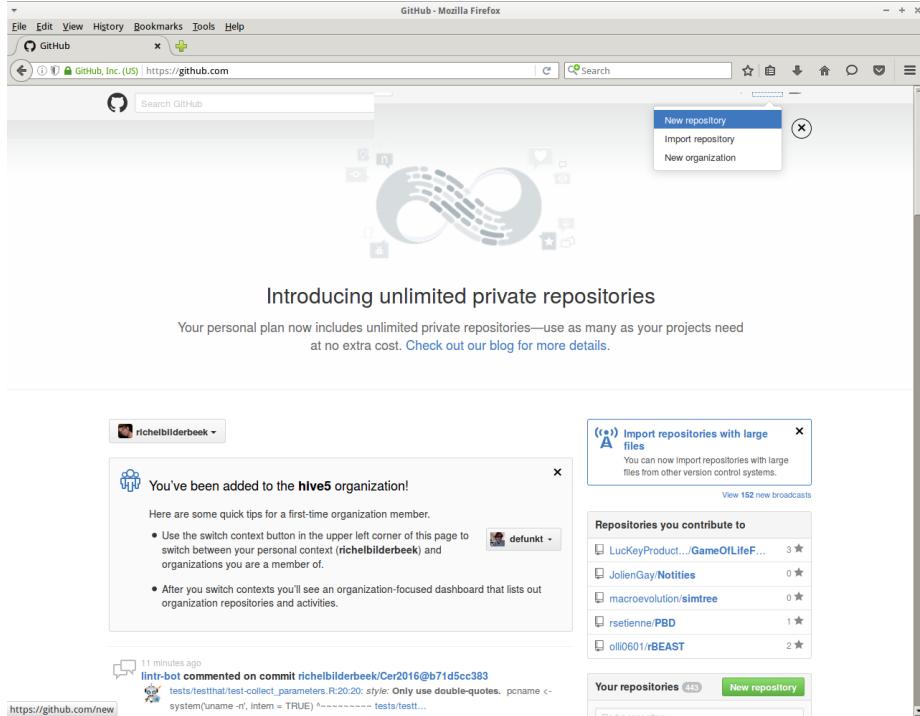


Figure 77: Maak een GitHub

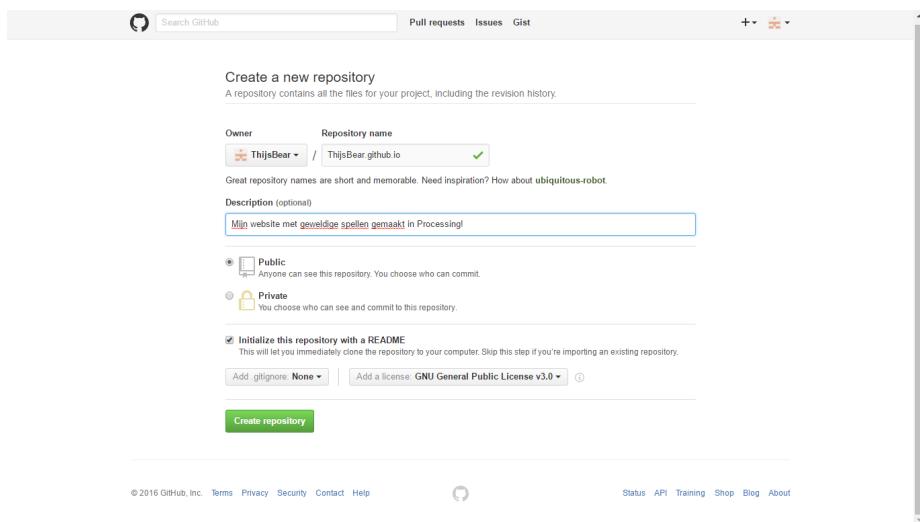
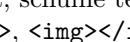


Figure 78: Maak GitHub Page repository

HTML staat voor HyperText Markup Language en is een opmaaktaal. Met een opmaaktaal kun je aangeven hoe tekst eruit moet komen te zien, een andere veel gebruikte opmaaktaal is Markdown (deze pagina is in Markdown geschreven).

Om tekstopmaak te doen in een programma zoals *Microsoft Word* heb je allerlei knoppen; dikgedrukte tekst, schuine tekst, plaatje, etc. In HTML heb je daar *tags* voor; ****, *<i></i>*, .

Een stukje HTML met een tag gaat altijd op deze manier:

```
<tag>(n)iets</tag>
```

- **<tag>** opent de tag, in plaats van **tag** staat hier het soort tag wat je wil gebruiken (bijvoorbeeld **h1** of **p**)
- **(n)iets** kan een heleboel zijn, zoals teks, nog meer tags, of helemaal niks
- **</tag>** sluit de tag, hier moet in plaats van **tag** hetzelfde als in de openende tag staan

Een HTML-bestand begint altijd met **<!doctype html>**, dit vertelt een programma wat het bestand opent dat het in HTML geschreven is. De rest van het bestand staat in één **html** tag, die weer is opgedeeld in één **head** tag en één **body** tag.

In de **head** tag staat informatie *over* de webpagina, in de **body** tag staat wat er *op* de webpagina komt.

Een stukje HTML kan er bijvoorbeeld zo uit zien:

```
<body>
  <h1>Hello World</h1>
  <p>This is <b>my</b> website!</p>
</body>
```

De homepage maken

Elke website is een map met bestanden, alleen staat de map niet op jouw computer. Een repository is ook een map, en de GitHub Page repository is ook jouw website. De homepage van elke website is het bestand **index.html**, die gaan we dus maken!

- Ga naar je GitHub Page repository
- Klik op **New file**
- Noem het nieuwe bestand **index.html**
- Begin het bestand met **<!doctype html>**
- Open op een nieuwe regel een **html** tag met **<html>**
- Open op een nieuwe regel een **head** tag met **<head>**
- Open op een nieuwe regel een **title** tag met **<title>**
- Typ hier de titel van je webpagina, bijvoorbeeld **Hello World**

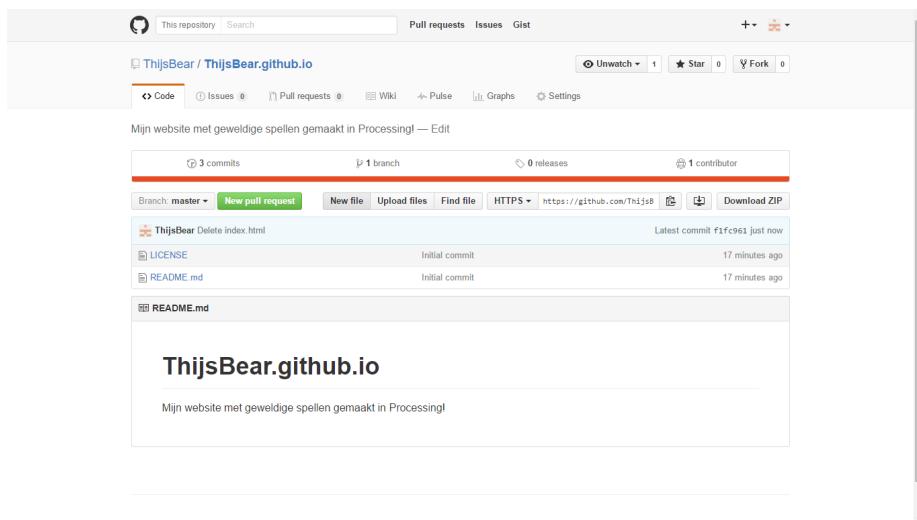


Figure 79: De GitHub Page repository

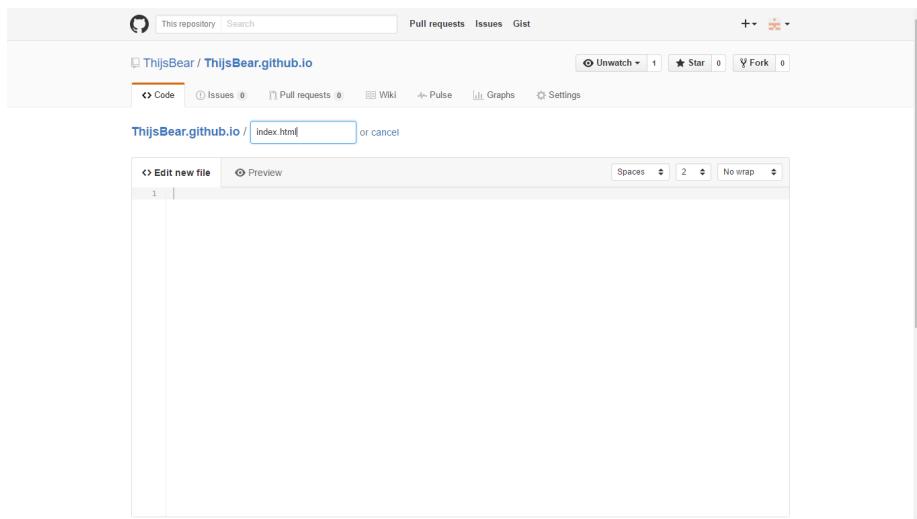
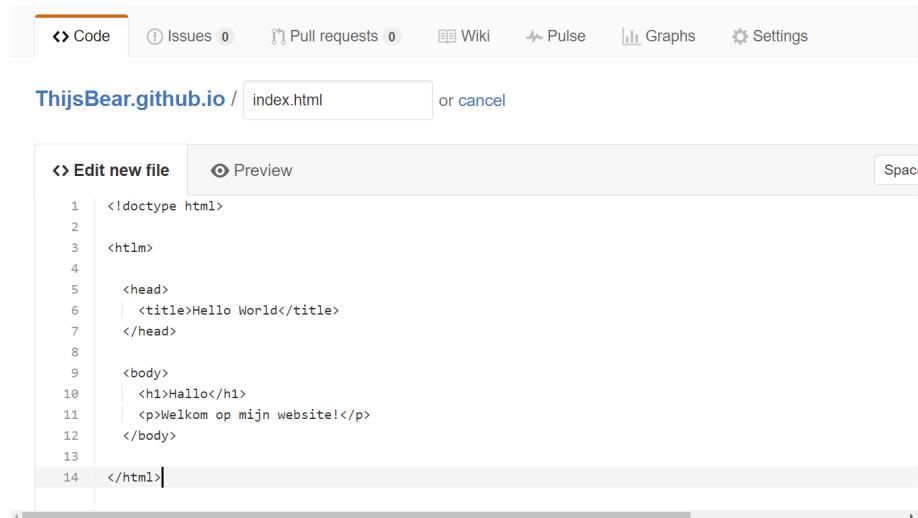


Figure 80: Maak index.html

- Sluit de `title` tag met `</title>`
- Sluit op een nieuwe regel de `head` tag met `</head>`
- Open op een nieuwe regel een `body` tag met `<body>`
- Open op een nieuwe regel een `h1` tag met `<h1>`
- Typ hier een kop voor je webpagina, bijvoorbeeld `Hallo`
- Sluit de `h1` tag met `</h1>`
- Open op een nieuwe regel een `p` tag met `<p>`
- Type hier een stukje tekst voor je webpagina, bijvoorbeeld `Welkom op mijn website`
- Sluit de `p` tag met `</p>`
- Sluit op een nieuwe regel de `body` tag met `</body>`
- Sluit op een nieuwe regel de `html` tag met `</html>`

Als het goed is ziet je bestand er nu zo uit:



The screenshot shows a GitHub repository interface. At the top, there are navigation links: Code, Issues 0, Pull requests 0, Wiki, Pulse, Graphs, and Settings. Below this, the repository name is shown as `ThijsBear.github.io / index.html`. A search bar contains the text "index.html" and includes "or cancel" and "Search" buttons. The main area is titled "Edit new file" and shows the following code:

```

1 <!doctype html>
2
3 <html>
4
5   <head>
6     <title>Hello World</title>
7   </head>
8
9   <body>
10    <h1>Hallo</h1>
11    <p>Welkom op mijn website!</p>
12  </body>
13
14 </html>

```

Figure 81: index.html

- Klik onderaan op de groene `Commit new file` knop
- Ga in je browser naar je GitHub Page en bewonder je eigen website!
- Ga terug naar je GitHub Page repository
- Klik bij `index.html` op het padloodje om het bestand te bewerken
- Maak een paar veranderingen en klik op `Commit changes`
- Bewonder nu je eigen gepersonaliseerde website en bekijk ook eens die van je klasgenoten!

Als je nog geen GitHub account hebt, doe dan eerst deze les

Als je nog geen GitHub Page hebt, doe dan eerst deze les

Processing.js

Processing.js is een JavaScript wat een Processing sketch op een webpagina kan runnen. Deze les gaan we een Processing sketch op onze GitHub Page zetten.

Met Processing.js kunnen mensen jou spel spelen, kunstwerk bewonderen of programma runnen op jouw GitHub Page. De code voor het spel, kunstwerk of programma staat op een GitHub repository.

Processing.js downloaden

Processing.js is een JavaScript bestand. Om het te gebruiken moeten we het eerst downloaden. * Ga in een webbrowser naar www.processingjs.org

Je komt nu op de homepage

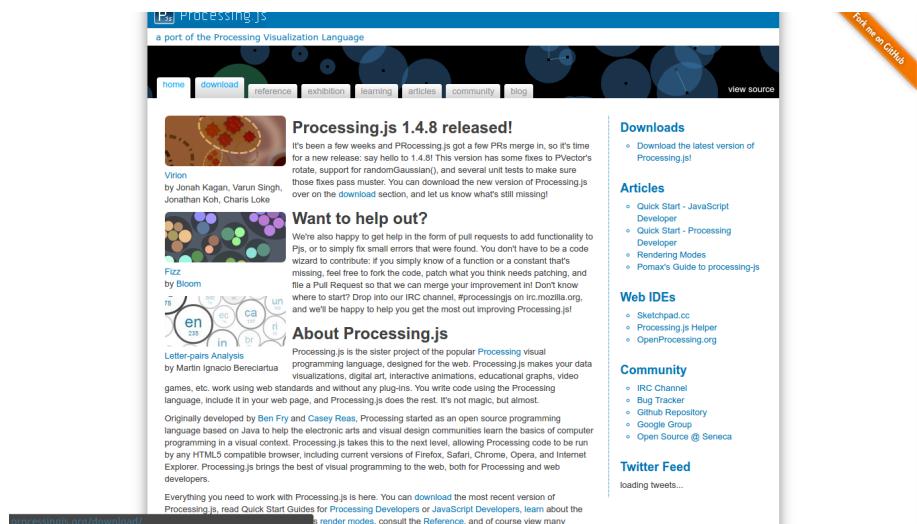


Figure 82: ProcessingJS homepage

- Klik bovenin op **download**

Je komt nu op de download pagina

- Klik met de rechter muisknop op **processing.js**
- Klik op **Link opslaan als...** (in het Engels: **Save link as...**)
- Kies een plek op je computer waar je het bestand zo terug kan vinden (Tip: Bureaublad of Desktop is vaak makkelijk te vinden!)
- Ga naar je GitHub Page repository
- Klik op **Upload file**

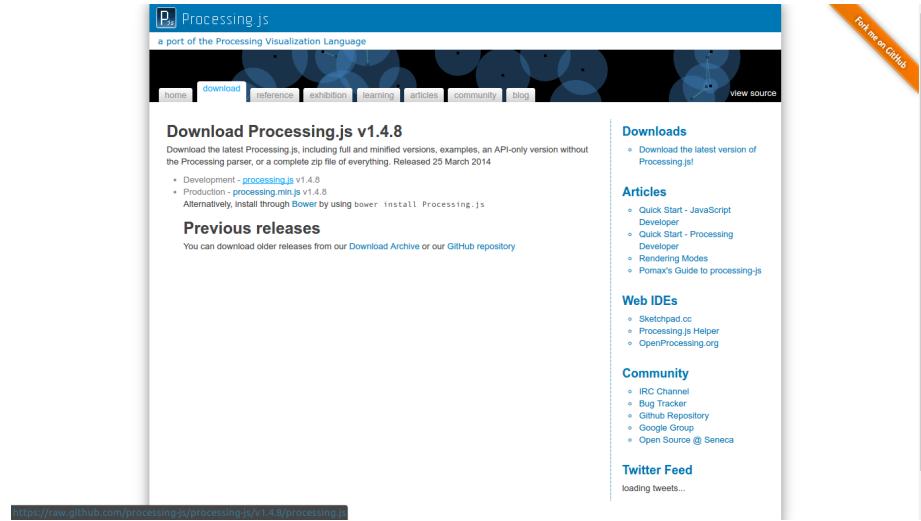


Figure 83: ProcessingJS download

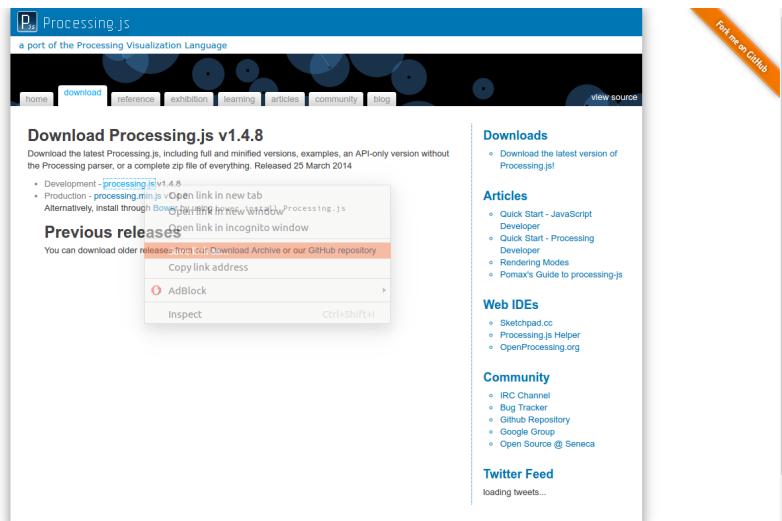


Figure 84: Link opslaan als

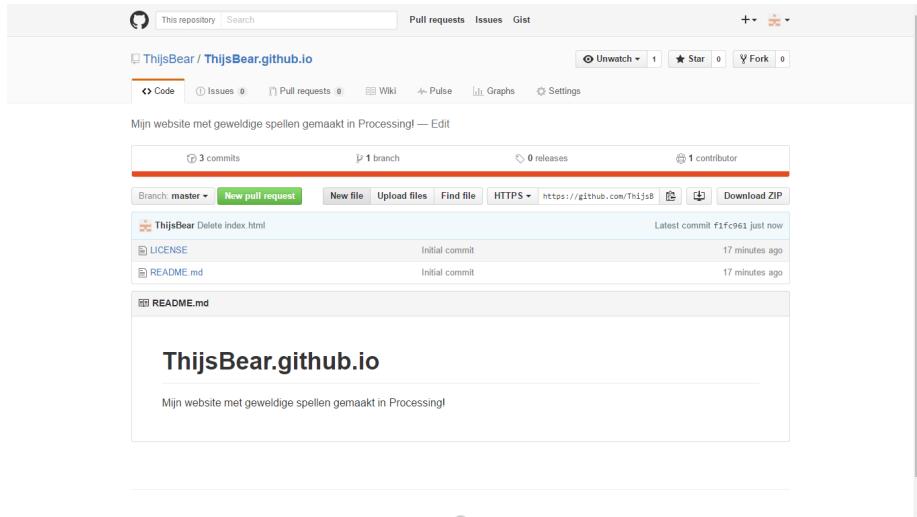


Figure 85: GitHub Page repository

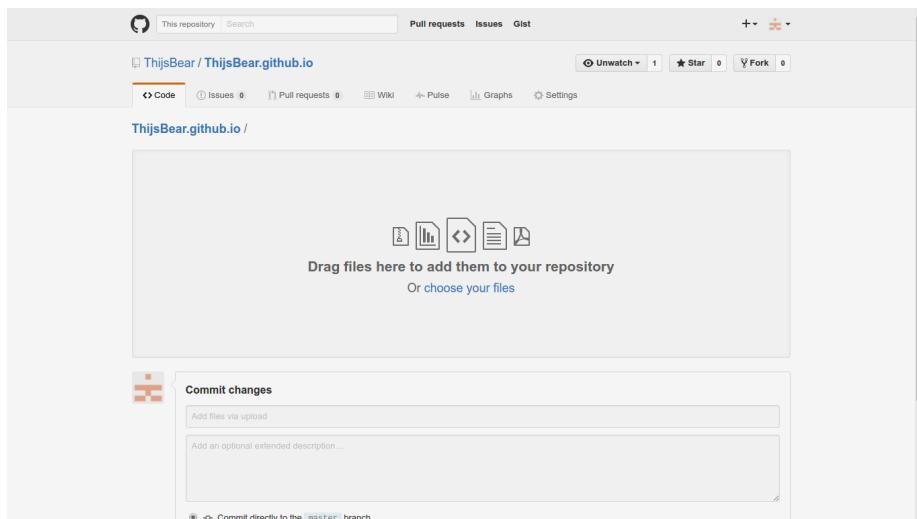


Figure 86: GitHub upload file

- Sleep `processing.js` dit scherm in of klik op `select your files` en dubbelklik op `processing.js`
- Klik onderaan op de groene knop `Commit changes`

Processing sketch uploaden

- Zoek een Processing sketch op je computer om op je website te zetten (Tip: kies een simpele sketch)
- Als je in de sketch map kijkt staat daar een `.pde` bestand, deze hebben we straks nodig
- Ga naar je GitHub Page repository

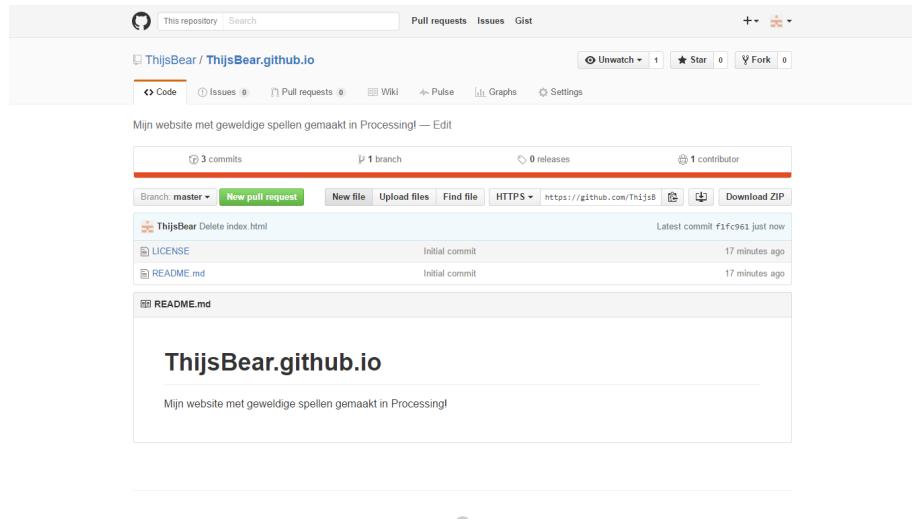


Figure 87: GitHub Page repository

- Klik op `Upload file`
- Sleep het `.pde` bestand dit scherm in of klik op `select your files` en dubbelklik op het `.pde` bestand
- Klik onderaan op de groene knop `Commit changes`

Nu ga je de sketch op je homepage zetten

- Ga naar je GitHub Page repository
- Klik op `index.html`
- Klik op het plusje rechts boven
- Open een nieuwe regel in head (dus tussen `<head>` en `</head>`)
- Typ hier deze regel over: `<script src="processing.js"></script>`
- Open een nieuwe regel in body (dus tussen `<body>` en `</body>`)
- Typ hier deze regel over: `<canvas data-processing-sources="voorbeeld.pde"></canvas>`

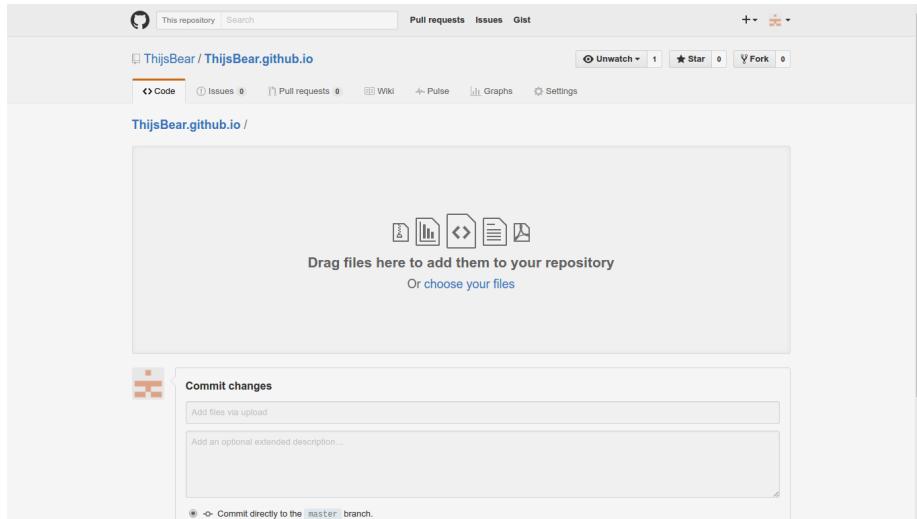


Figure 88: GitHub upload file

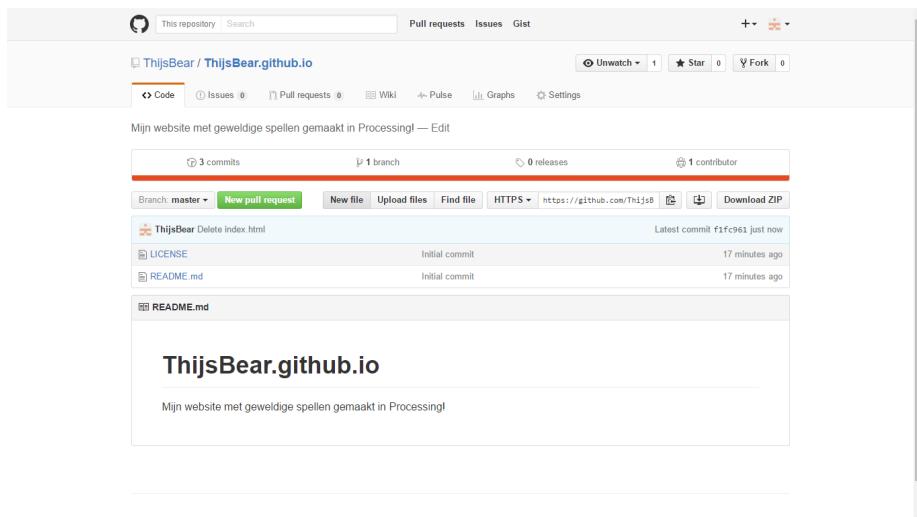
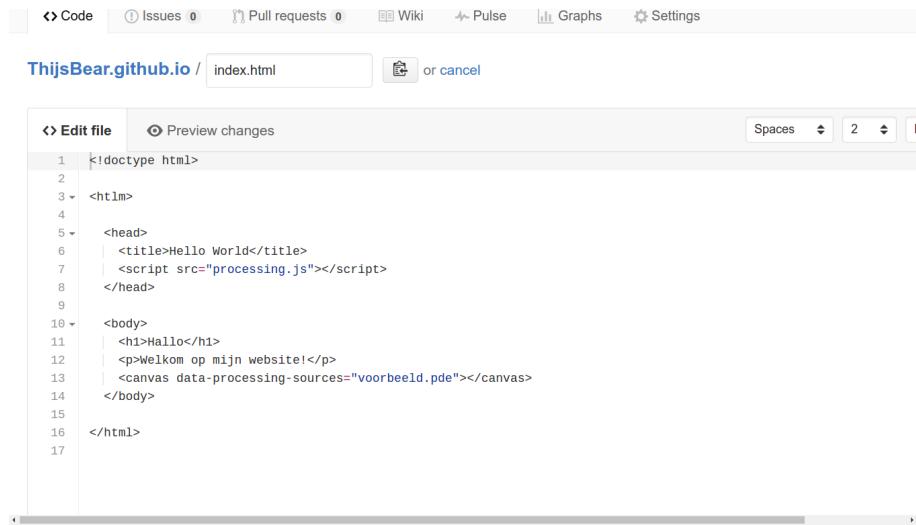


Figure 89: GitHub Page repository

- Vervang nu **voorbeeld** voor de naam van je .pde bestand

Het bestand moet er nu ongeveer zo uit zien:



The screenshot shows a GitHub repository interface with the URL "ThijsBear.github.io" and the file "index.html". The code editor displays the following HTML code:

```

1 <!doctype html>
2
3 <html>
4
5 <head>
6   <title>Hello World</title>
7   <script src="processing.js"></script>
8 </head>
9
10 <body>
11   <h1>Hallo</h1>
12   <p>Welkom op mijn website!</p>
13   <canvas data-processing-sources="voorbeeld.pde"></canvas>
14 </body>
15
16 </html>
17

```

Figure 90: index

- Klik onderaan op de groene knop **Commit changes**
- Ga naar je website en bewonder je werk!

GitHub chat

In deze les gaan we de chatroom van de cursus in.

Wat is een chatroom?

Er zijn meer soorten chatrooms. In deze les gebruiken we een IRC chatroom. IRC bestaat sinds 1986 en wordt nog veel gebruikt. Elke hackerspace heeft wel een eigen IRC chatroom (dit noemen we ook een 'IRC kanaal'). Een IRC chatroom is net een echt gesprek met een groepje mensen: je weet niet wat er is gezegd voordat je met ze sprak, en je weet niet wat ze zeggen als je weer weggaat. Er zijn wel IRC chatrooms die wel de gesprekken bewaren.

Gedragsregels

Een paar van <https://wiki.debian.org/IRC/Netiquette>:

- Wees behulpzaam
- Praat over het thema, in ons geval programmeren
- Probeer mee te doen met het gesprek, in plaats van nieuwe gesprekken te starten
- Stel jezelf voor als je mee doet in een gesprek en nieuw bent
- ‘Don’t feed the trolls’. Help de mensen die de sfeer verpesten niet hierbij: blijf altijd vriendelijk
- Heb plezier

Website

Ga naar de GitHub van de cursus, <https://github.com/richelbilderbeek/Dojo>. Klik op irc: #dojoGroningen

IRC inloggen

- Vul bij **Nickname** je bijnaam in
- Klik op **I am not a robot**

Some krijg je een vraag, zoals hier:

Beantwoord de vraag.

Klik dan op **Connect**

IRC server login

Eerst komt er een scherm met veel tekst:

Wacht tot dit scherm voorbij is

IRC chatroom

Je komt nu in de chatroom:

Rechts kun je een lijst zien van iedereen die in deze kamer aanwezig is.

Onderin kun je je tekst typen:

De tekst verschijnt bij iedereen meteen op hun scherm.

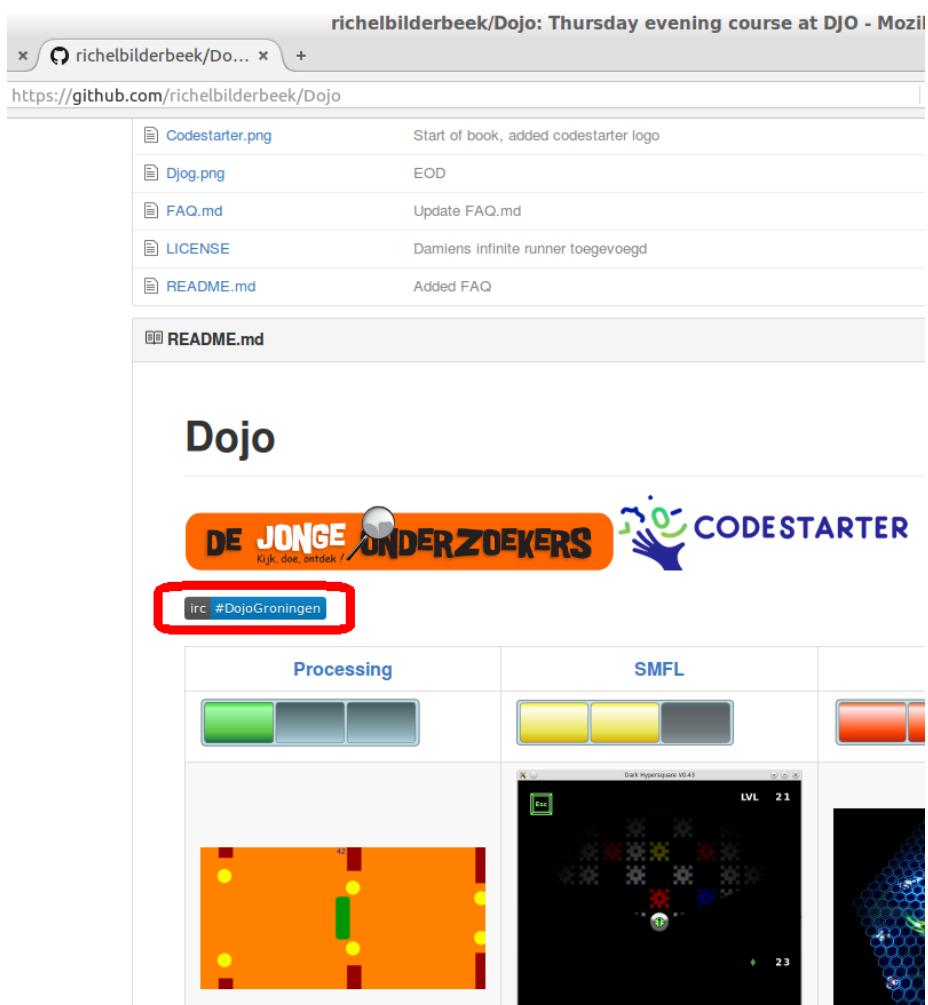


Figure 91: GitHub.png

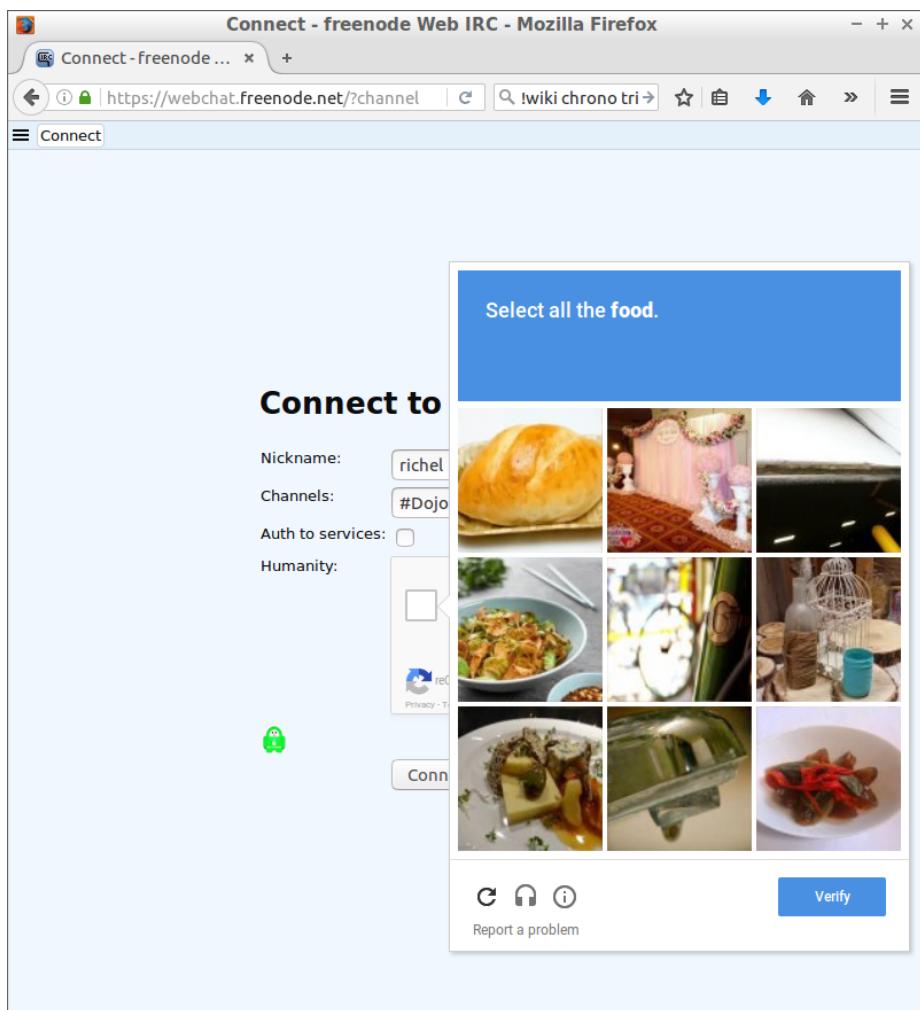


Figure 92: IrcInlog.png

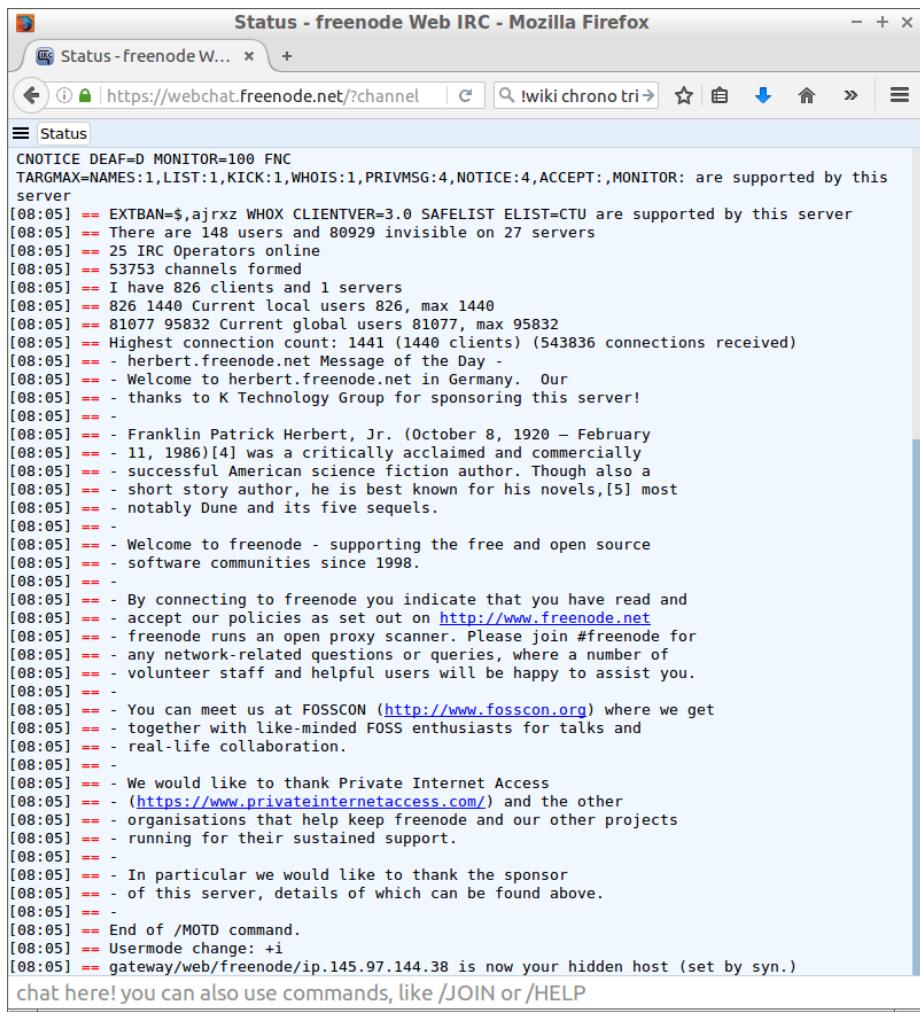


Figure 93: IrcServerMessages.png

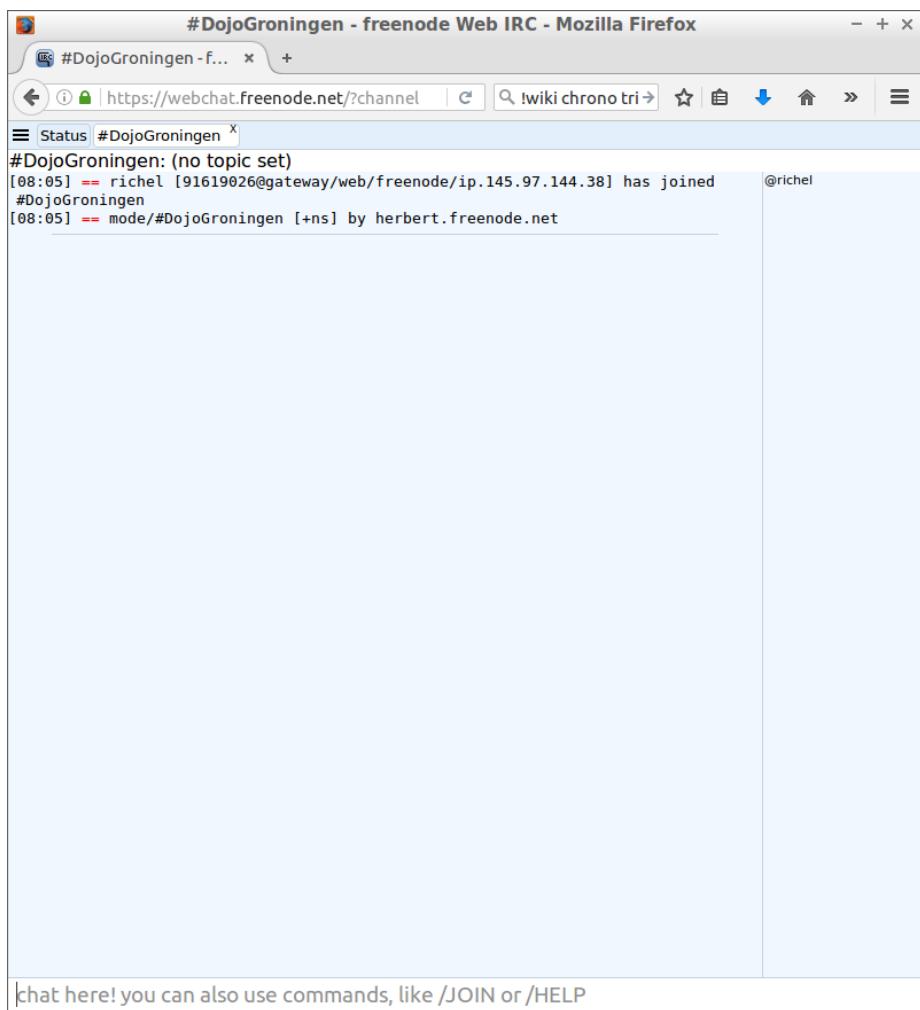


Figure 94: IrcChatroom1.png

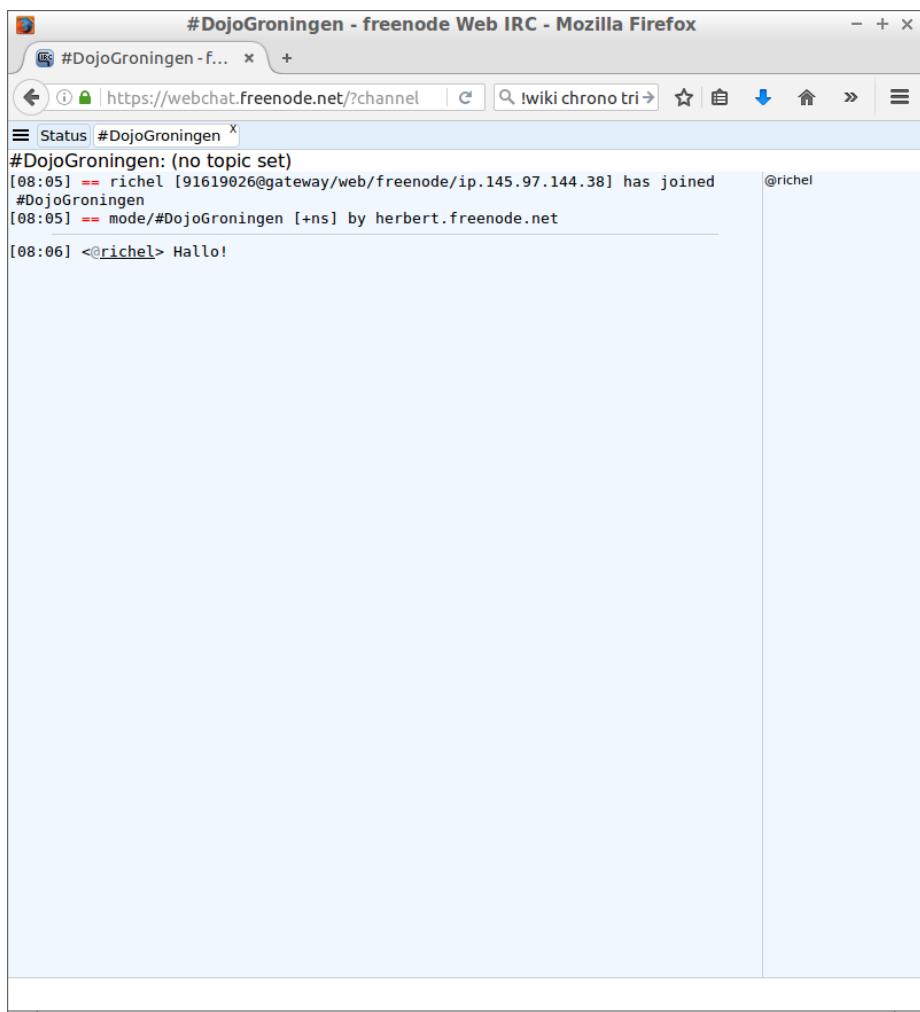


Figure 95: IrcChatroom2.png

git, GitHub samenwerken

Samenwerken is lastig, zeker met programmeren: de ander (of jij) maakt misschien dingen stuk en/of kwijt.

In deze les gaan we leren samenwerken met git en GitHub. Dit doen we (nog) niet met code, maar met een Nederlandse tekst.

git, GitHub

git houdt de geschiedenis van de tekst/code bij. Je kunt ook terug in de geschiedenis gaan. Als je per ongeluk een bestand hebt gewist, kun je deze weer terughalen.

Sprookje

In deze les gaan we samen een sprookje maken. Hier staat de begintekst:

```
# Sprookje

## 1

Heel erg lang geleden ...

## 2

## 3

## 4

## 5

## 6

## 7

... en ze leefden nog lang en gelukkig.
```

Het sprookje is nog erg saai. Samen kunnen we er een leuk verhaal van maken.

De GitHub op je computer zetten

Om het sprookje op je computer te zetten, start je (onder Windows) Git Bash, onder Linux een terminal.

Type dan het volgende:

```
git clone https://github.com/richelbilderbeek/Sprookje
```

Er is nu een map gemaakt, genaamd **Sprookje** met daarin het bestand **README.md**. **README.md** bevat de tekst van het sprookje en kun je openen in Kladblok/Wordpad/leafpad of een andere teksteditor.

Samenwerken

Verander de tekst in **README.md** en save.

Doe dan in Git Bash:

```
git pull
```

```
git add --all :/
```

```
git commit -m "Verbetering"
```

```
git push
```

- **git pull**: update de GitHub
- **git add --all :/**: bekijk alles wat veranderd is
- **git commit -m "Verbetering"**: benoem de veranderingen **Verbetering**
- **git push**: publiceer je veranderingen

Merge conflicts

Soms krijg je ‘Merge conflicts’. Dit gebeurt als **git** niet weet, waar welke veranderingen horen.

Stel, je hebt dit:

```
## 1
```

```
## 2
```

Nu maakt Jantje er dit van:

```
## 1
```

```
Er woonde een gevvaarlijke draak in de berg ten oosten van het dorpje.
```

```
## 2
```

En maakt Truus er tegelijkertijd dit van:

```
## 1
```

```
De prinses was op zoek naar de knapste ridder.
```

```
## 2
```

`git` kan dit niet samenvoegen, omdat de volgorde tussen de zinnen onduidelijk is.

Degene die het merge conflict krijgt, krijgt een `README.md` zoals dit:

```
## 1
<<<<<<<<<HEAD: 372546042986273465283736726872
Er woonde een gevaarlijke draak in de berg ten oosten van het dorpje.
=====
De prinses was op zoek naar de knapste ridder.
TAIL>>>>>>>>: 623483754623592736429736239874
```

2

De persoon zal dan zelf de zinnen moeten samenvoegen:

1

```
Er woonde een gevaarlijke draak in de berg ten oosten van het dorpje.
Ten westen van het dorpje was een kasteel met een prinses.
De prinses was op zoek naar de knapste ridder.
```

2

Dan weer:

```
git pull
git add --all :/
git commit -m "Verbetering"
git push
```

Voorwoord

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.

(C) Dojo Groningen 2016

The screenshot shows the Processing 3.2.3 IDE interface. The title bar reads "Voorwoord | Processing 3.2.3". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The toolbar has a play/pause button and a Java dropdown. The code editor window is titled "Voorwoord" and contains the following text:

```
1 //*****
2      PROCESSING
3 ****
4
5 Dit is het Processing boek
6 van de Dojo.
7
8 Processing is een programmeertaal.
9
10 Dit boek leert je die taal.
11
12 Over dit boek
13 -----
14
15 Dit boek heeft een CC-BY-NC-SA
16 licentie.
17
18 (C) Dojo Groningen 2016 
19 */
20
```

Below the code editor, there is a banner for "DE JONGE ONDERZOEKERS" with the tagline "Kijk, doe, ontdek!" and a "CODESTARTER" logo. The status bar at the bottom shows "Done saving." and tabs for "Console" and "Errors".

Figure 96: Voorblad



Figure 97: Het logo van De Jonge Onderzoekers



Figure 98: Het logo van Codestarter



Figure 99: De licentie van dit boek