Processing

Voorwoord

Dit is het Processing boek van de Dojo. Processing is een programmeertaal. Dit boek leert je die programmeertaal.

Over dit boek

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

Daarom staat dit boek op een GitHub. Om precies te zijn, op https://github.com/richelbilderbeek/Dojo . Hierdoor kan iedereen die dit boek te slordig vindt minder slordig maken.

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

Figure 1: Voorblad

Figure 2: Het logo van De Jonge Onderzoekers

Figure 3: Het logo van Codestarter

Figure 4: De licensie van dit boek

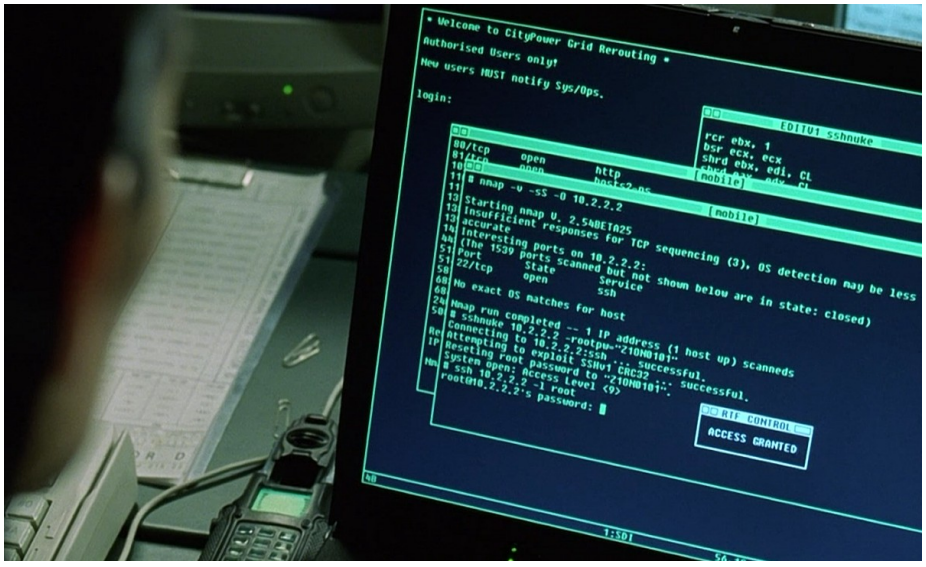

Figure 5: Hackers werken met een terminal

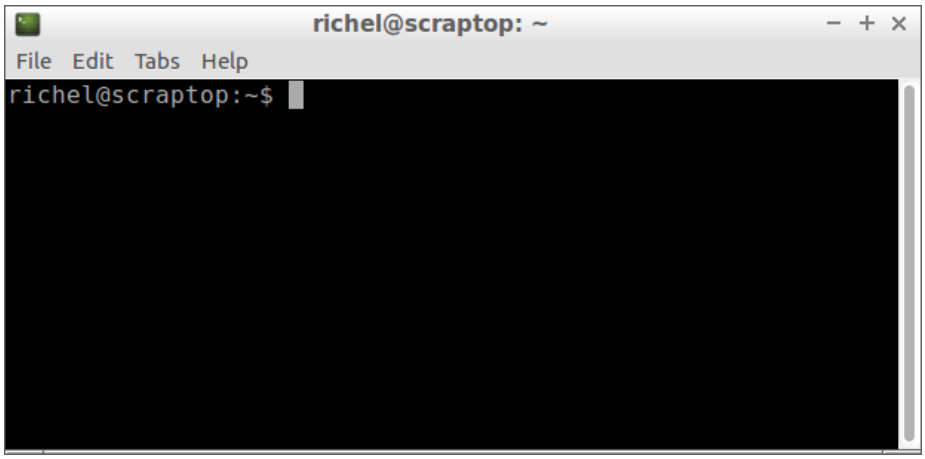

Figure 6: Een terminal

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

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

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

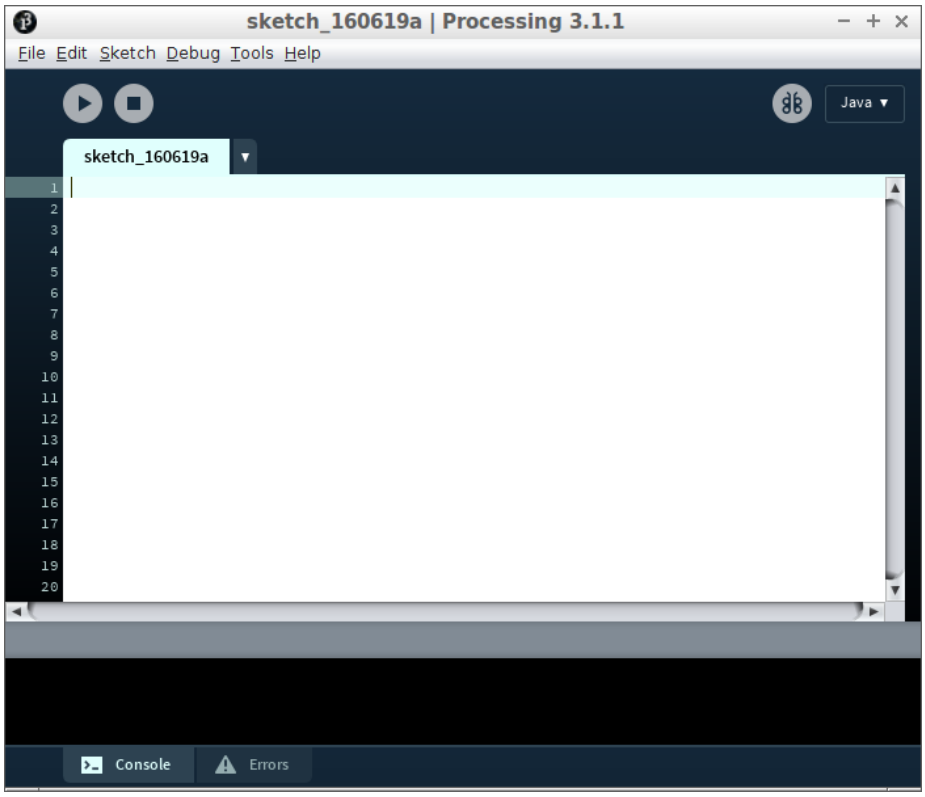

Figure 7: Processing zonder code

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:

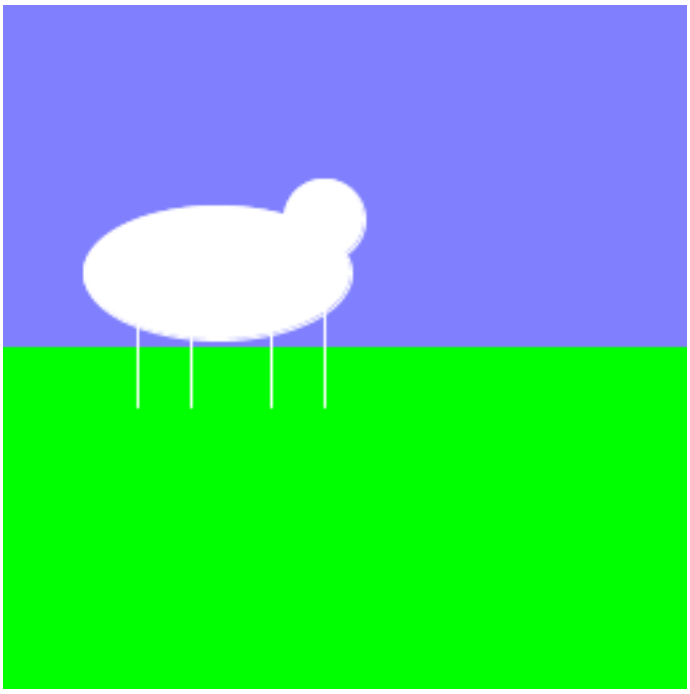

Figure 8: Wit Schaap

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 hypthese te controleren!

Functies les 3: Poten

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

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

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

We beginnen met deze code:

```
int schapen;

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

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

```
  println("Voor " + schapen + " schapen, zijn er " + schapen * 4 + " poten!");

}

void draw(){

}
```

Deze code print dit naar de console:

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

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

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

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

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

De functie wordt dan:

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

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

Opdrachten

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

Figure 9: Voorblad

Voorwoord

Figure 10: Het logo van De Jonge Onderzoekers

Figure 11: Het logo van Codestarter

Dit is het Processing boek van de Dojo.

Processing is een programmeertaal.

Dit boek leert je die programmeertaal.

Over dit boek