Processing handleiding

Inhoud

1 Inleiding

Processing is een programmeeromgeving die is gebaseerd op de programmeertaal Java. Java is een van de meest gebruikte programmeertalen in de wereld.

Een voordeel van Processing is dat het programmeren in Java makkelijker is gemaakt.

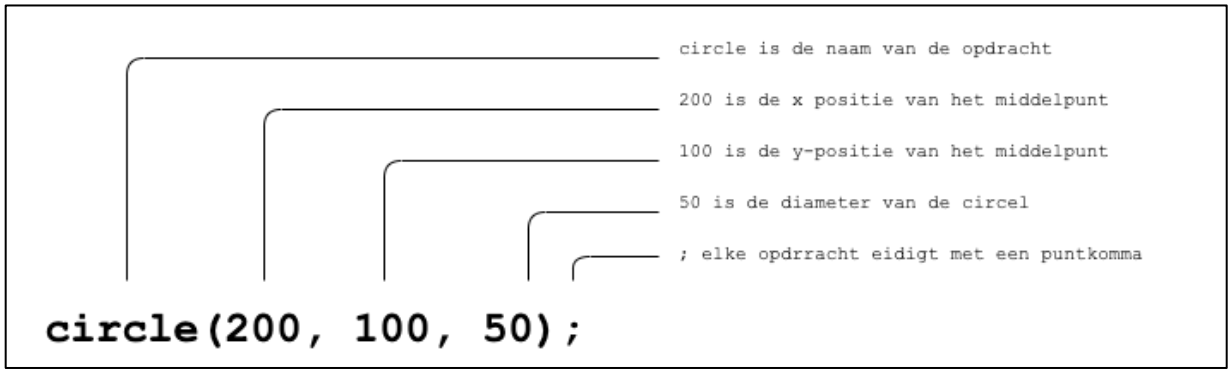
Processing kan je downloaden van de website https://processing.org/

2 Opdrachten

In een programma gebruik je opdrachten om aan de computer te vertellen wat hij moet doen.

Een opdracht bestaat uit een naam en parameters. De parameters worden tussen ronde haakjes gegeven en geven extra informatie over wat er moet gebeuren. Een opdracht eindigt met een puntkomma.

In het voorbeeld hieronder wordt de opdracht gegeven om een cirkel te tekenen op positie (200, 100) met een diameter van 50 pixels. De waarden 200, 100, 50 zijn hier de parameters.



Hieronder staan een paar opdrachten uitgelegd, maar Processing kent er nog veel meer.

Opdracht	Uitleg	Voorbeeld
`circle(x, y, d);`	Teken een cirkel plaats ***(x, y)*** met een diameter van ***d***.	`circle(200, 100, 50);`
`rect(x, y, width, height)`	Reken een rechthoek op plaats ***(x,y)*** met een breedte van ***width*** en een hoogte van ***height***.	`rect(10, 10, 40, 50);`

Het is ook mogelijk om zelf opdrachten te maken. Dit worden functies genoemd. Functies worden later in deze handleiding uitgelegd.

3 Expressies

4 Structuurelementen

In dit hoofdstuk worden een aantal structuurelementen uitgelegd waarmee je ervoor kan zorgen dat een deel van je programma meer dan 1 keer worden uitgevoerd, of dat een deel van een programma soms wordt overgeslagen.

In structuurelementen worden voorwaarden gebruikt om aan te geven of een deel van je programma wel of niet moet worden uitgevoerd. Een voorwaarde kan op een bepaald moment waar of onwaar zijn. Bijvoorbeeld:

```
mouseX < 200
```

Deze voorwaarde is waar als de x-positie van de muis kleiner is dan 200, maar de voorwaarde is onwaar als de x-positie van de muis groter is of gelijk is aan 200. Het is niet erg als je dit nu nog niet helemaal begrijpt, dit wordt straks vanzelf duidelijk.

4.1 for

Met het structuurelement ***for*** kun je een aantal regels code meerdere keren achter elkaar uitvoeren.

Een for ziet er als volgt uit:

```
for (beginopdracht; voorwaarde; herhaalopdracht) {
    // code…
}
```

Beginopdracht: Wordt 1 keer uitgevoerd aan het begin.

Voorwaarde: Als de voorwaarde waar is wordt de code binnen {} nog een keer uitgevoerd.

Herhaalopdracht: Wordt telkens uitgevoerd aan het einde van de code binnen {}

In het onderstaande voorbeeld wordt de variabele ***i*** aan het begin op ***0*** gezet. De code tussen ***{}*** wordt net zo vaak uitgevoerd als dat de voorwaarde ***i < 200*** waar is. Daarna wordt telkens de variabele ***i*** met ***1*** verhoogt.

Het resultaat is dat de code dus 200 keer wordt uitgevoerd.

```
for (int i = 0; i < 200; i = i + 1) {
    // code...
}
```

Bijvoorbeeld:

```
for (int i = 0; i < 5; i = i + 1) {
    println("Dit is herhaling nummer " + i);
}
```

Print:

```
Dit is herhaling nummer 0
Dit is herhaling nummer 1
Dit is herhaling nummer 2
Dit is herhaling nummer 3
Dit is herhaling nummer 4
```

4.2 if en else

Met een if statement kun je zorgen dat een aantal regels code alleen wordt aangeroepen als aan een voorwaarde wordt voldaan. Vervang 'voorwaarde' met een voorwaarde die alleen `true` of `false` kan geven.

```
if (voorwaarde) {
    code...
} else {
    andere code...
}
```

Bijvoorbeeld:

```
if (afstand > 10) {
    println("Ik ben er nog lang niet");
} else {
    Println("Ik ben er bijna");
}
```

Print *'Ik ben er nog lang niet'* als de afstand groter is dan 10. De code print *'Ik ben er bijna'* als de kleiner is of gelijk is aan 10.

4.3 while

Met een while loop wordt een aantal regels code meerdere keren achter elkaar uitroepen, zolang er aan de voorwaarde wordt voldaan. Vervang 'voorwaarde' met de voorwaarde.

```
while (voorwaarde) {
    code...
}
```

Bijvoorbeeld:

```
int hoeveelheidAppels = 0;
while (hoeveelheidAppels < 3) {
    println("We hebben nog steeds niet genoeg appels");
    hoeveelheidAppels++;
}
```

Print:

```
We hebben nog steeds niet genoeg appels
We hebben nog steeds niet genoeg appels
We hebben nog steeds niet genoeg appels
```

7 Informatie

7.1 width en height

width en height zijn twee variabelen die de breedte en hoogte van je programma aangeven in pixels.

7.2 millis()

Geeft aan hoe lang het programma al draait in milliseconden.

7.3 mouseX en mouseY

mouseX en mouseY geven de huidige positie van de muis aan.

7.4 mousePressed

De mousePressed variabele geeft `true` als minstens een muisknop is ingedrukt en `false` als dit niet het geval is.

7.5 mouseButton

De mouseButton variabele geeft aan welke muisknop voor het laatst is ingedrukt. Dit kan zijn `LEFT` voor de linkermuisknop, `RIGHT` voor de rechtermuisknop en `CENTER` voor de knop van het scrol wieltje.

7.6 keyPressed

De keyPressed variabele geeft `true` als minstens een toets op het toetsenbord is ingedrukt en `false` als dit niet het geval is.

7.7 key

De key variabele geeft aan welke toets voor het laatst is ingedrukt.