Processing

Basis

Functie	Beschrijving	Voorbeeld
size(breedte, hoogte)	Nieuw venster openen waarin we beelden kunnen tonen.	size(960, 540)
save("bestandsnaam.extensie")	Het resulterende beeld opslaan als een bestand	save("image.png");
print(string)	Een string 'printen' naar de console	print("Multec is cool"); print(x);
println	Een string + nieuwe lijn 'printen' naar de console	println("Multec is cool"); println(x);

Een variabele declareren

datatype naam = waarde

int	***(gehele getallen)***	*bv. int x = 1;*
float	***(decimale getallen)***	*bv. float snelheid = 120.80;*
String	***(tekst)***	*bv. string naam = "Dieter";*
boolean	***(true/false)***	*bv. boolean isSmart = true;*

Een variabele kan enkel aangesproken worden vanuit het instructieblok waarin het wordt gedeclareerd en alle hierin geneste instructieblokken (= het *bereik*)

Een functie declareren

terugkeertype naam(parameters) {
instructies/implementatie
}

- terugkeertype: het datatype van de waarde die de functie na uitvoeren teruggeeft
  - → functies die geen waarde teruggeven: terugkeertype *void*, d.m.v. *return* (geen haakjes)
  - → max. 1 terugkeerwaarde kan uitgevoerd worden d.m.v. *return()*
    (meerdere *return*-instructies mag, maar er wordt er maar één uitgevoerd)
- naam: naam waarmee de functie later opgeroepen kan worden
  - → zelfde voorwaarden als variabelen:
    - eerste karakter moet een letter of een liggend streepje zijn
    - daaropvolgende karakters moeten een letter, cijfer of een liggend streepje zijn
- parameters: variabelen die dienen als plaatshouders voor argumenten, gedeclareerd in de functiedefinitie
- argumenten: de concrete waarden die we als parameters willen gebruiken en die bij het uitvoeren van de functie meegestuurd kunnen worden
- implementatie: de reeks van instructies waarmee een functie een bepaalde functionaliteit realiseert

Nut van functies:

- een reeks instructies hergebruiken binnen een programma
- functies als modulaire bouwstenen waarmee we complexe functionaliteit kunnen opbouwen
- abstractie maken van implementatiedetails van de functie om zich te kunnen concentreren op de complexe functionaliteit

→ *Er kunnen ≠ definities aan een functie gegeven worden, met ≠ aantal en datatype van parameters*

Een array initialiseren

- Alle waarden in een array moeten het zelfde datatype hebben
- De lengte van de array kunnen we op voorhand bepalen maar kan daarna niet meer gewijzigd worden
- De positie van een element in een array noemen we de index van dat element in de array.
- In een array begint de index te tellen bij 0

Een array initialiseren: *type*[] *naam* = new *type*[*lengte*];
Een waarde uit een array opvragen: *naam*[*index*];
Een waarde in een array plaatsen: *naam*[*index*] = *waarde;*

Datatype van de waarden	Declaratie en initialisatie	Initiele waarden
int	`int[] reeks = new int[10];`	0
float	`float[] reeks = new float[10];`	0.0
boolean	`boolean[] reeks = new boolean[10];`	false
String	`String[] reeks = new String[10];`	null
PImage	`PImage[] reeks = new PImage[10];`	null
PFont	`PFont[] reeks = new PFont[10];`	null

Operatoren

	naam	operatoren	
1.	**unaire** *operatoren die een bewerking uitvoeren op 1 getal*	**-** **!**	unaire negatie-operator unaire ontkennings-operator
2.	**multiplicatieve**	***** **/** **%**	vermenigvuldigen delen modulus *(restwaarde v.e. deling)*
3.	**additieve**	**+** **-**	optellen aftrekken
4.	**relationele** *twee uitdrukkingen met elkaar vergelijken*	**<** **>** **<=** **>=**	kleiner dan groter dan kleiner dan of gelijk aan groter dan of gelijk aan
5.	**gelijkheids** twee uitdrukkingen op gelijkheid testen	**==** **!=**	gelijk aan niet gelijk aan
6.	**toekenning** *de nieuwe waarde van een variabele berekenen op basis van de vorige waarde, in één expressie* *bv. v += b => v = v + b*	**=** **+=** **-=** ***=** **/=** **%=** **++** **--**	toekenning x = x + y x = x - y x = x * y x = x / y x = x % y x = x + 1 x = x - 1

Vormen tekenen

Functie	Beschrijving	Voorbeeld
point(x, y)	Een punt tekenen.	point(100, 100)
line(x1, y1, x2, y2)	Een lijn tekenen tussen 2 punten.	line(200, 100, 400, 300)
quad(x1, y1, x2, y2, x3, y3, x4, y4)	Een vierhoek tekenen.	quad(100, 200, 500, 100, 500, 200, 100, 300)
rect(x, y, breedte, hoogte) (als de standaard rect-modus actief is)	Een rechthoek tekenen.	rect(100, 100, 100, 100)
rectMode(parameter)	De rect-modus aanpassen.	CORNER 1,2 = linkerbovenhoek 3,4 = breedte, hoogte CORNERS 1,2 = linkerbovenhoek 3,4 = tegenovergestelde hoek RADIUS 1,2 = middelpunt 3,4 = helft van de breedte/hoogte CENTER 1,2 = middelpunt 3,4 = breedte, hoogte
ellipse(x, y, breedte, hoogte) (als de standaard ellipse-modus actief is)	Ellipsen en cirkels tekenen.	ellipse(300,150,100,100)
ellipseMode(parameter)	De ellipse-modus aanpassen.	CENTER 1,2 = middelpunt 3,4 = breedte, hoogte RADIUS 1,2 = middelpunt 3,4 = helft van de breedte/hoogte CORNER 1,2 = linkerbovenhoek 3,4 = breedte, hoogte CORNERS 1,2 = linkerbovenhoek 3,4 = tegenovergestelde hoek

Vertex

= complexere vormen tekenen d.m.v. een reeks verbonden punten te specificeren

```
beginShape(soort);     // POINTS, LINES, TRIANGLES, QUADS, etc. of leeg
vertex(x1, y1);
vertex(x2, y2);
vertex(x3, y3);
    ...
endShape(mode);        // CLOSE (begin- en eindpunt verbinden) of leeg
```

Kleuren

Functie	Beschrijving	Voorbeeld

background(grijswaarde)
background(rood, groen, blauw)
background(rood, groen, blauw, alpha)
	De achtergrondkleur van het scherm instellen	background(125);
		background(125,230,180);
		background(125,230,180,120);

stroke(grijswaarde)
stroke(rood, groen, blauw)
stroke(rood, groen, blauw, alpha)
	Bepalen in welke kleur lijnen en omlijningen getekend zullen worden.	stroke(125);
		stroke(125,230,180);
		stroke(125,230,180,120);

noStroke()	Geen omlijningen tekenen.	

fill(grijswaarde)
fill(rood, groen, blauw)
fill(rood, groen, blauw, alpha)
	Bepalen in welke kleur o.a. vierhoeken en ellipsen worden gevuld.	fill(125);
		fill(125,230,180);
		fill(125,230,180,120);

noFill()	Vormen niet opvullen met een vul-kleur.	

colorMode(mode)
colorMode(mode, max)
colorMode(mode, max1, max2, max3)
colorMode(mode, max1, max2, max3, maxA)
	Bepalen op welke manier kleurwaarden worden geïnterpreteerd:
	- RGB of HSB
	- bereik van de kleurwaarde (standaard: 0 – 255)
		colorMode(RGB);
		colorMode(RGB, 100);
		colorMode(HSB, 360, 100, 100);
		colorMode(HSB, 360, 100, 100, 1);

Invoer

Functie	Beschrijving	Voorbeeld

mousePressed()	Functie die opgeroepen wordt elke keer een muisknop wordt ingedrukt.	`void mousePressed() {`
		`  println(“KLIK!”);`
		`}`

mouseClicked()	Functie die opgeroepen wordt elke keer een muisknop wordt ingedrukt en losgelaten.	`void mouseClicked() {`
		`  println(“KLIK!”);`
		`}`

Interpolatie

= de positie van een waarde binnen een bereik op een ander bereik projecteren (met dezelfde verhouding),

Functie: map(v, a, b, a', b')

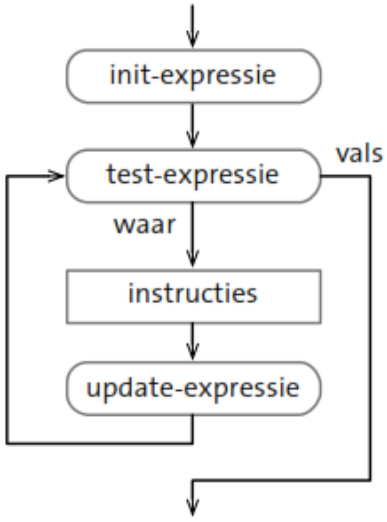
De functie neemt 5 argumenten:

1. de waarde die geprojecteerd wordt,
2. de beginwaarde van het bereik van waaruit geprojecteerd wordt,
3. de eindwaarde van het bereik van waaruit geprojecteerd wordt,
4. de beginwaarde van het bereik waarop geprojecteerd wordt,

de eindwaarde van het bereik waarop geprojecteerd wordt.

Iteratiestructuren

Functie	Beschrijving	Voorbeeld

for (init-expressie; test-expressie; update-expressie) {
instructies
}
	init-expressie
	een declaratie en initialisatie van een variabele die je zal gebruiken als teller tijdens de iteratie.
	
	test-expressie
	moet na evaluatie een Booleaanse waarde (*true* of *false*) als resultaat geven, indien true wordt overgegaan naar de instructies.
	
	instructies
	Een reeks instructies die elke keer opnieuw wordt uitgevoerd
	
	update-expressie
	(meestal) een iteratie van de variabele die in de init-expressie werd gedeclareerd
		`for (int i = 0; i <= 3; i++)`
		`{`
		`  println(i);`
		`}`
		 

while (expressie) {
instructies ;
}
	Blijft de instructiereeks herhalen zo lang de expressie true is	`int i = 0;`
		`while (i < 80) {`
		`  line(30, i, 80, i);`
		`  i = i + 5;`
		`}`

Selectiestructuren

= bepaalde instructies in een programma enkel uitvoeren indien aan bepaalde condities wordt voldaan

Instructies uitvoeren indien aan een voorwaarde voldaan wordt

```
if (testexpressie) {
  if-instructies
}
```

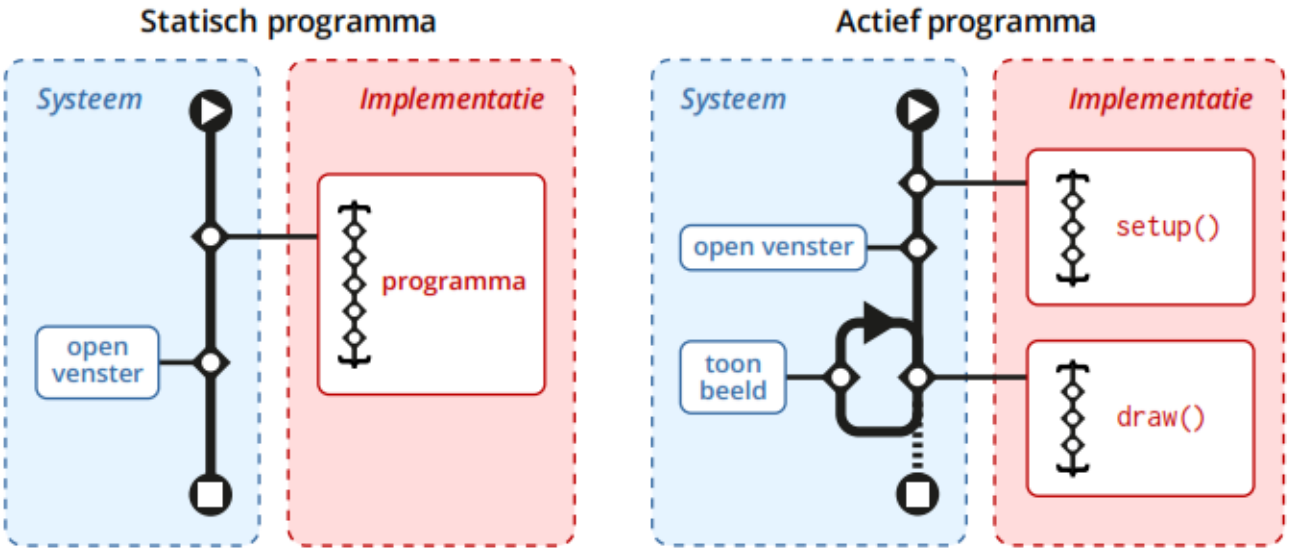
Instructies uitvoeren indien wel of niet aan een voorwaarde voldaan wordt

```
if (testexpressie) {
  if-instructies
}
else {
  else-instructies
}
```

Instructies uitvoeren indien wel of niet aan één of meerdere voorwaarden voldaan wordt

```
if (testexpressie) {                // Moet steeds één voorzien zijn
  if-instructies                    // (verplicht)
}
else if (testexpressie) {           // Mag meerdere keren voorkomen
  else-if-instructies
}
else {
  else-instructies                  // Mag slechts éénmaal voorkomen
}                                   // (niet verplicht)
```

Actieve programma's



1. Globale variabelen declareren

2. *setup*-functie

= initialisatie (bevat onder andere de *size*-functie)

```
void setup() {
   setup-instructies
}
```

3. *draw*-functie

= de update-loop (iteratie)

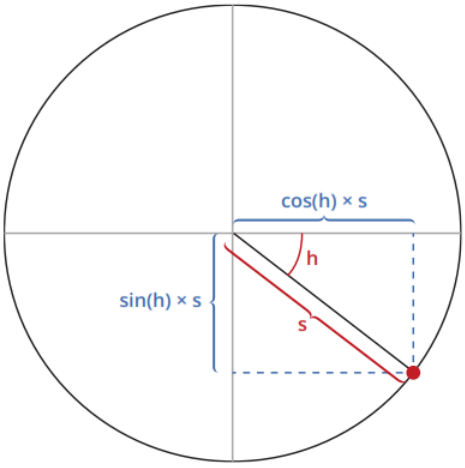
```
void draw() {
   draw-instructies
}
```

Functie	Beschrijving
frameCount	Systeemvariabele die het aantal iteraties van de update-loop bijhoudt
exit()	Het programma beëindigen
noLoop()	De update-loop beëindigen
loop()	De update-loop hervatten nadat hij gestopt werd
frameRate	Systeemvariabele die de *frame rate* bijhoudt
frameRate()	De frequentie waarmee Processing de update-loop herhaalt (standaard: 60)
saveFrame("bestandsnaam.extensie")	Huidige frame opslaan (*hash-tekens #* in de bestandsnaam worden vervangen door het nummer van de frame)

Goniometrie

Om de x/y-positie van een bepaald punt op een cirkel te bekomen maken we gebruik van goniometrie:

```
float x = cx + cos(h) * s;
float y = cy + sin(h) * s;
```



gebruik makend van volgende variabelen:

`cx` = de x-positie van het middelpunt van de cirkel

`cy` = de y-positie van het middelpunt van de cirkel

`s` = de straal van de cirkel

`h` = de hoek van de beoogde punt op de cirkel, uitgedrukt in radialen

Functie	Beschrijving
PI	Ingebouwde constante die π voorstelt
TWO_PI	Ingebouwde constante die 2 keer π voorstelt
HALF_PI	Ingebouwde constante die π/2 voorstelt
QUARTER_PI	Ingebouwde constante die π/4 voorstelt
radians(*graden*)	Converteert graden naar radialen

Transformaties

- worden uitgevoerd op het hele coördinatenstelsel, niet alleen getekende vormen
- meerdere transformaties die mekaar volgen worden geïnterpreteerd volgens het op dat moment geldende coördinatenstelsel (m.a.w. het originele coördinatenstelsel wordt niet hersteld)
- transformaties vertrekken vanuit het nulpunt van het op dat moment geldende coördinatenstelsel
- transformaties (en meer bepaald *pushMatrix* en *popMatrix*) werken met een first-in-last-out-datastructuur: het eerste element op de stack wordt er het laatste afgehaald, het laatste element wordt er als eerste afgehaald (bv. een queue/wachtrij)

Functie	Beschrijving	Voorbeeld
translate(*x, y*)	Het coördinatenstelsel horizontaal en/of verticaal verplaatsen.	translate(50, 50);
rotate(*radialen*)	Het coördinatenstelsel roteren volgens een bep. hoek.	rotate(PI / 4);
scale(*verhouding*) **scale(*x, y*)**	Het coördinatenstelsel schalen volgens een bep. verhouding.	scale(2);
pushMatrix()	De huidige toestand van het coördinatenstelsel op de *stack* plaatsen (meestal in het begin van een iteratie)	
popMatrix	De toestand die bovenaan de *stack* staat terug toepassen op het coördinatenstelsel (meestal op het einde van een iteratie)	