

Vectoren

Steven Bronsveld

February 23, 2019

1 Gegevens

Uiterlijke inleverdatum: **Datum 1**

1.1 Links

- [Github.com/StevenBrons](https://github.com/StevenBrons)
- <https://natureofcode.com/book>
- <http://hello.processing.org/editor/>

2 Leerdoelen

- `x,y` coördinaten om kunnen zetten in de `PVector` objecten.
- Gebruik kunnen maken van de volgende `PVector` methods: `add(PVector p)`, `sub(PVector p)`, `mult(float amount)`, `rotate(float angle)`

3 Vectoren

Omdat het onhandig is om telkens twee argumenten mee te moeten geven voor een positie op het scherm `int x`, `int y` en we een betere manier nodig hebben om met coördinaten om te gaan bestaat er in Processing de `PVector` class.

3.1 [optioneel] Vectoren in de wiskunde

Een vector is een verzameling van meerdere variabelen. Wij zullen ons alleen maar bezig houden met 2 dimensionale vectoren van `x,y` coördinaten. Een vector wordt als volgt genoteerd:

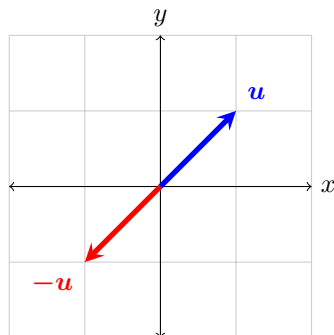
$$\vec{v} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Er zijn een paar rekenregels, die erg voor de hand liggen als je bedenkt dat een vector gewoon een verzameling van twee coördinaten is:

$$\begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a + c \\ b + d \end{pmatrix}$$

$$a * \begin{pmatrix} b \\ c \end{pmatrix} = \begin{pmatrix} a * b \\ a * c \end{pmatrix}$$

De tweede rekenregel heet *scalair vermenigvuldiging*. Dit geeft het uitrekken of inkrimpen van een vector weer. Dit is makkelijker te zien als we de vectoren als pijltjes (of natuurkundige krachten) tekenen:



3.1.1 Opdrachten

1. Teken de optelling van $\begin{pmatrix} 2 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ 3 \end{pmatrix}$
2. Bereken $2 * ((3 * \vec{a}) + \vec{b})$ met $\vec{a} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ en $\vec{b} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$
3. Bepaal het midden tussen \vec{a} en \vec{b} . (We zoeken dus een algemene formule voor het midden tussen twee vectoren).
4. Bereken de vector op $\frac{2}{3}$ afstand tussen \vec{a} en \vec{b} (Wederom zoeken we dus een algemene formule).

3.2 PVector

Processing heeft de class **PVector**, met daarin een heleboel handige methods, zie <https://processing.org/reference/PVector.html>

```
1      void setup() {
2          PVector v1 = new PVector(3,2);
3          PVector v2 = v1.copy();
4          v1.add(v2);
5          v2.sub(new PVector(1,1));
6          v1.mult(3);
7          drawDot(v1);
8          drawDot(v2);
9      }
10
11     void drawDot(PVector v) {
12         circle(v.x,v.y,5);
13     }
```

Let op! De oorsprong (0,0) zit bij computers links boven, en niet links onder zoals bij de meeste wiskundige grafieken! De y-as is als het ware gespiegeld!

Op welke coördinaten tekent dit stukje code een stip? Schrijf je antwoord in een *comment* van je sketch:

3.3 Polygoon

Een gelijkzijdige polygoon of veelhoek is een figuur met **n** hoeken en lijnstukken van gelijke lengte. Voor **n** = 3 is dit een *driehoek*, voor **n** = 4 is dit een *vierkant*, voor **n** = 5 is dit een *pentagon* en voor **n** = 17 is dit een *heptadecagoon*. Maak de volgende functie:

```
1      void polygon(PVector center, int radius, int n) {
2      }
```

Deze functie moet een polygoon van **int n** hoeken tekenen met een straal van **int radius**. Maak gebruik van vectoren en gebruik de **rotate(float angle)** method.

Let op! Een hoek wordt niet in graden uitgedrukt maar in radialen, dit betekent dat één cirkel (dus 360 graden) gelijk is aan $2 * \pi$, ofwel: $2 * \text{PI}$.

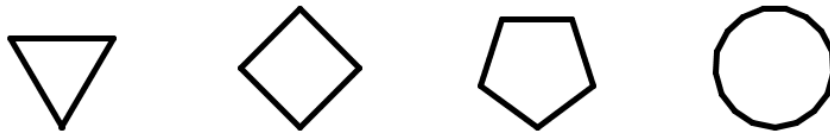


Figure 1: Polygonen met $n = 3, 4, 5$ en 17

4 Middelloodlijn

Maak de volgende functie aan:

```
1 void bisector(PVector p1, PVector p2) {
2 }
```

Deze functie moet de middelloodlijn tekenen van de twee functies met de lengte gelijk aan de afstand tussen de twee functies (zie figuur 4). **Tip:** Probeer eerst op papier een aantal middelloodlijnen te tekenen en probeer vervolgens om de twee punten waartussen de lijn getrokken moet worden te vinden.

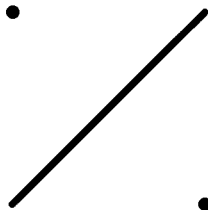


Figure 2: Middelloodlijn, de twee stippen geven $p1$ en $p2$ aan

5 Koch's Curve gen0

Maak de functie

```
1 void kochCurve(PVector p1, PVector p2) {
2 }
```

Deze functie tekent de volgende figuur tussen $p1$ en $p2$: Hierbij ligt $m2$ midden tussen $p1$ en $p2$.

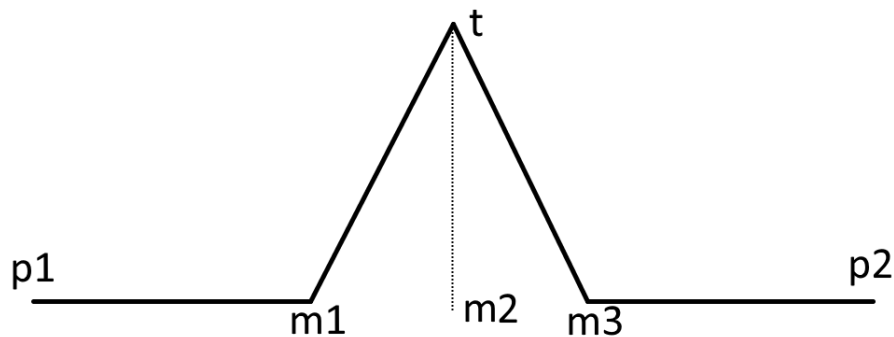


Figure 3: Koch's curve gen 0

De afstand tussen p1 en m1 is even groot als de afstand tussen m2 en t.

m1 ligt op $\frac{1}{3}$ afstand van p1 naar p2.

m3 ligt op $\frac{2}{3}$ afstand van p1 naar p2.

Tip: Dit is een redelijk complexe opdracht. Teken dit eerst uit, en probeer al voordat je begint met coderen te weten *wat* je precies gaat coderen. Maak gebruik van de `dist` `rotate` en `mult` `PVector` methods.

6 [Bonus] Hilbert's Curve

Maak de volgende functie:

```

1  void hilbertCurve(PVector p1,PVector p2) {
2  }

```

Die de volgende figuur tekent:

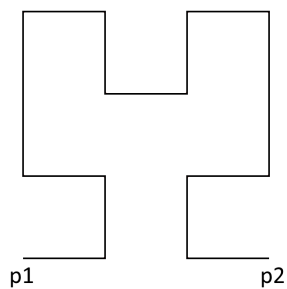


Figure 4: Hilbert's Curve

7 Inleveren

Als je klaar bent met de hele opdracht kun je deze naar je *repository* pushen.