1 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)`

2 Uitleg

- Nature Of Code Chapter 1
  `https://natureofcode.com/book/chapter-1-vectors/`
  Alleen **1.1, 1.2, 1.3, 1.4, 1.5**
- Vectors - The Nature of Code
  `https://www.youtube.com/watch?v=mWJkvxQXIa8\&list=PLRqwX-V7Uu6ZwSmtE13iJBcoI-r4y7iEc`
  Alleen **1.1, 1.2, 1.3, 1.4**

3 Voorbeelden

1. `star`
2. `square`
3. `forest`
4. `flower`

4 Opdrachten

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.

4.1 [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:

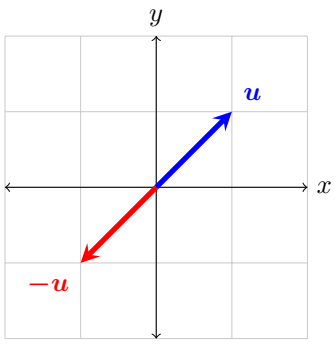
$$\vec{v} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

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 *scalaire 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:



4.1.1 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).

4.2 [Optioneel] PVector

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

```
void setup() {
  PVector v1 = new PVector(3,2);
  PVector v2 = v1.copy();
  v1.add(v2);
  v2.sub(new PVector(1,1));
  v1.mult(3);
  drawDot(v1);
  drawDot(v2);
}

void drawDot(PVector v) {
  circle(v.x,v.y,5);
}
```

Opmerking

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:

4.3 [Optioneel] Vectoren gebruiken

Nu je hebt geleerd wat een vector is wil je natuurlijk deze coole vectoren voor alles gebruiken! Helaas accepteren de Processing functies geen vectoren, alleen `x, y` coördinaten. Maak de volgende 3 functies af, de functie `myLine` is al gegeven.

```
void myLine(PVector v1, PVector v2) {
    line(v1.x,v1.y,v2.x,v2.y);
}

void myCircle(PVector v1, int r) {
    // TODO
}

void myTriangle(PVector v1, PVector v2, PVector v3) {
    // TODO
}
```

4.4 Een driehoek

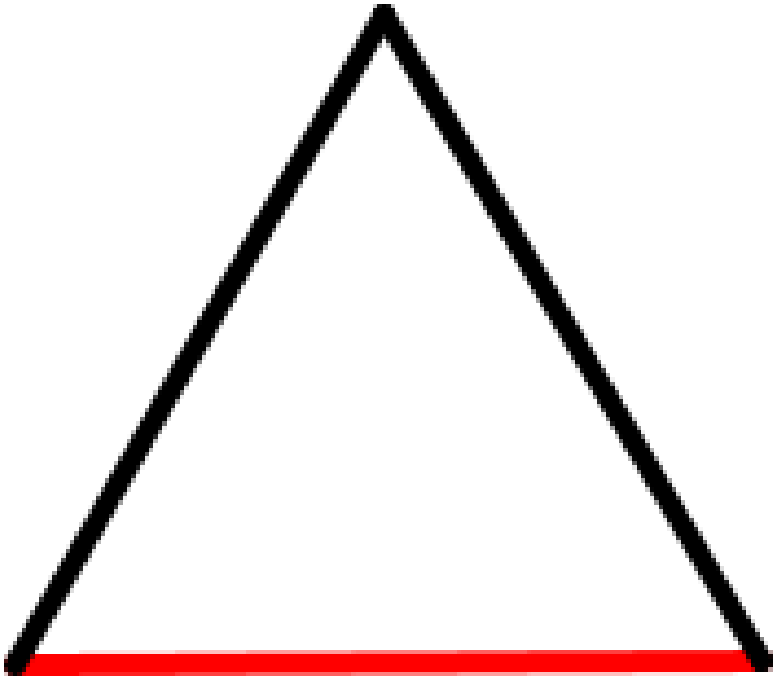
Maak de functie

```
void betterTriangle(PVector p1, PVector side) {
    // TODO
}
```

Hierbij is `p1` één van de hoekpunten en `side` één van de zijden.

Tip

Kijk naar voorbeeld `square`



/4/4

Figure 1: Een triangle met de vector `side` rood gekleurd

4.5 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:

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

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.

Opmerking

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 * PI`.

4.6 Middelloodlijn

Maak de volgende functie aan:

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

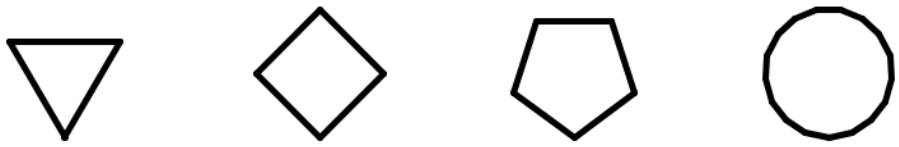


Figure 2: Polygonen met n = 3, 4, 5 en 17

Deze functie moet de middelloodlijn tekenen van de twee functies met de lengte gelijk aan de afstand tussen de twee functies (zie figuur **??**).

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.

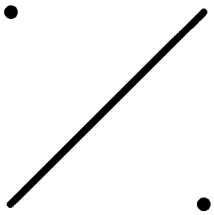


Figure 3: Middelloodlijn, de twee stippen geven `p1` en `p2` aan