1 Leerdoelen

- Begrip voor het concept recursie
- Het coderen van recursieve functies
- Het kunnen tekenen van simple fractels

2 Uitleg

- Fractals - The Nature of Code
  `https://www.youtube.com/watch?v=-wiverLQl1Q\&list=PLRqwX-V7Uu6bXUJvjnMWGU5SmjhI-OXef\&index=1`
  Alleen **8.1, 8.2, 8.3**
  Bij 8.3 wordt er gebruik gemaakt van een `ArrayList` *voor het maken van Koch's Curve. Dit hoef je (nog) niet te kunnen.*
- Chapter 8. Fractals
  `https://natureofcode.com/book/chapter-8-fractals/`
  Alleen **8.1, 8.2, 8.3, 8.4, 8.5**
  Bij 8.4 wordt er gebruik gemaakt van een `ArrayList` *voor het maken van Koch's Curve. Dit hoef je (nog) niet te kunnen.*

3 Opdrachten

3.1 [Optioneel] Recursieve functies

Recursieve functies zijn functies die een (makkelijkere) versie van zichzelf gebruiken voor het bereken van een antwoord. Kijk bijvoorbeeld naar:

$$f(n) = \begin{cases} 1 & \text{n} = 0 \\ 3 * \text{f(n-1)} & \text{n} > 0 \end{cases}$$

Deze functie geeft $f(n) = g^n$. Als we dit uitschrijven krijgen we: $f(4) = 3 * f(3) = 3 * 3 * f(2) = 3 * 3 * 3 * f(1) = 3 * 3 * 3 * 3 * f(0) = 3 * 3 * 3 * 3 * 1 = 81 = 3^4$

$$g(n) = \begin{cases} 1 & \text{n} = 0 \\ \text{n} * \text{g(n-1)} & \text{n} > 0 \end{cases}$$

Schrijf de uitwerking van $g(5)$ helemaal uit. Weet je ook welke functie g is?

$$h(n) = \begin{cases} 1 & \text{n} = 0 \\ 1 & \text{n} = 1 \\ \text{h(n - 1) + h(n - 2)} & \text{n} > 1 \end{cases}$$

Schrijf de uitwerking van $h(4)$ helemaal uit. Weet je ook welke functie h is?

3.2 Recursieve circles

Als een functie *zichzelf* aanroept noemen we dit **recursie**. Je kunt met recursie allerlei coole dingen tekenen. Een recursieve functie moet wel ooit stoppen, daarom geeft je vaak een getal mee `int n` wat het aantal **itteraties** geeft, ofwel, hoe vaak de functie aangeroepen wordt, of hoe "diep" de functie gaat.
Pas de onderstaande code aan zodat het resultaat lijkt op figuur **??**

```
void recursiveCircles(int n, PVector pos) {
  if (n > 0) {
    circle(pos.x,pos.y,n * 10);
    recursiveCircles(n - 1,pos);
  }
}
```

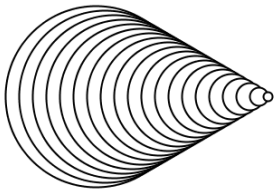

Figure 1: Recursief getekende cirkels

3.3 Maak een binary tree

In deze opdracht ga je een **binary tree** maken (zie figuur **??**). Een binary tree bestaat uit één lijn, genaamd de stam (trunk). Vanuit de top van deze stam "groeien" weer twee nieuwe stammen, alleen dan gedraaid onder een hoek. Maak de functie `binaryTree` af.

```
void binaryTree(int n,PVector pos,PVector trunk) {
  if (n > 0) {
    PVector tip = pos.copy().add(trunk);
    line(pos.x,pos.y,tip.x,tip.y);
  }
}
```

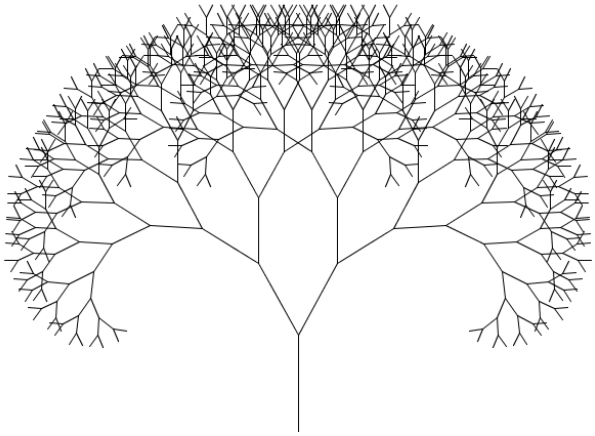

Figure 2: Een binary tree

3.4 Meer soorten trees!

Vervang de hoek die je gebruikt hebt bij de vorige opdracht door `(float) mouseX / width)`. Aanschouw de epische trees!

Opmerking

Zorg ervoor dat je de functie aanroept in de `draw` functie