Fractale du dragon

Eric BERTHOMIER

eric.berthomier@free.fr

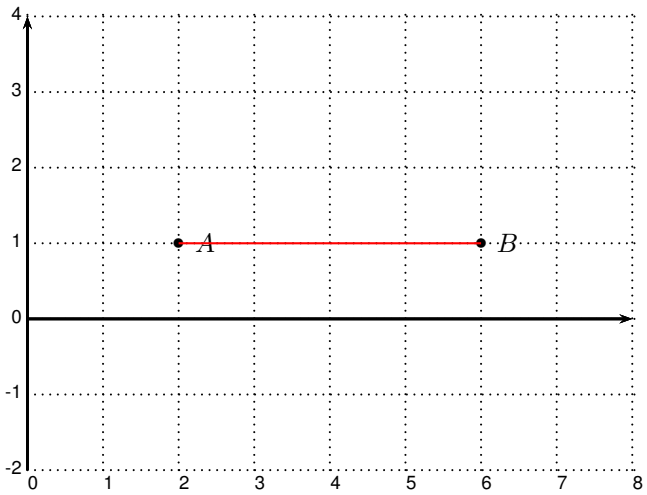
6 décembre 2023

La fractale du dragon est construite sur une construction géométrique de base : le triangle isocèle rectangle.

1 Rang n=0

Pour $n = 0$, la représentation de la fractale est un segment.

Soit $A(2, 1)$ $B(6, 1)$ les deux points associés à ce segment.
La fractale du dragon pour n=0 est donc :



Fractale du dragon - $n = 0$

2 Rang n=1

Pour $n = 1$, la représentation de la fractale se fait de la façon suivante :
— Trouver le point C pour que le triangle ACB soit un triangle rectangle isocèle en C.
— Tracer les segments AC et les segments CB

Remarque

✓ 2 points C sont possibles, en effet, il est possible de créer 2 triangles rectangles isocèles rectangles à partir d'un segment $[AB]$.

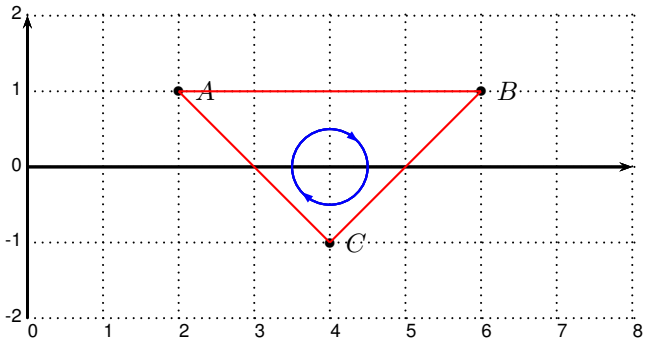
2.1 Triangle direct / indirect

Cette définition m'a été donné donnée par N_comme_Nul (invité) sur L'île des mathématiques.
Si tu te donnes une orientation (peu importe laquelle), si tu énumères les sommets d'un triangle ABC dans le "même sens" alors ton triangle est direct (cf. ABC) et indirect dans le cas contraire (cf. ACB)..

2.2 Démonstration mathématique

Cette démonstration m'a été donné donnée par lake sur L'île des mathématiques.

— C est l'image de B dans la similitude directe de centre A, d'angle $\frac{\pi}{4}$ et de rapport $\frac{1}{\sqrt{2}}$.



Direct / Indirect

— Similitude : Composition dans un ordre quelconque de la rotation de centre A et d'angle $\frac{\pi}{4}$ et de l'homothétie de centre A et de rapport $\frac{1}{\sqrt{2}}$

— En complexes, on écrit :

— $z_C - z_A = \frac{1}{\sqrt{2}} e^{\frac{i\pi}{4}} (z_B - z_A)$ ou encore :

— $z_C = \frac{1+i}{2}(z_B - z_A) + z_A \; x_C + iy_C = \frac{1+i}{2}[x_B - x_A + i(y_B - Y_A)] + x_A + iy_A$

— Après développement et séparation des parties réelles et imaginaires :

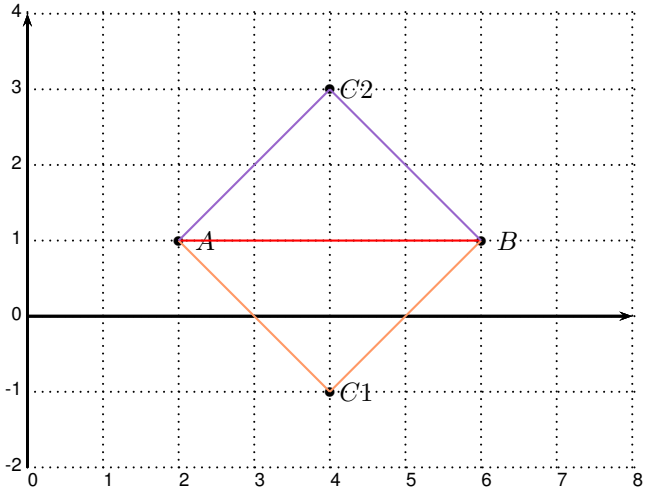
$$x_C + iy_C = \underbrace{\frac{x_A + x_B}{2} - \frac{y_B - y_A}{2}}_{x_C} + i \left(\underbrace{\frac{y_A + y_B}{2} + \frac{x_B - x_A}{2}}_{y_C} \right)$$

On obtient donc les coordonnées du point C pour que ACB soit un triangle rectangle isocèle **direct** de la façon suivante :

$$\begin{cases} x_C = \dfrac{(x_A + x_B)}{2} + \dfrac{(y_B - y_A)}{2} \\ y_C = \dfrac{(y_A + y_B)}{2} - \dfrac{(x_B - x_A)}{2} \end{cases}$$

On obtient donc les coordonnées du point C pour que ACB soit un triangle rectangle isocèle **indirect** de la façon suivante :

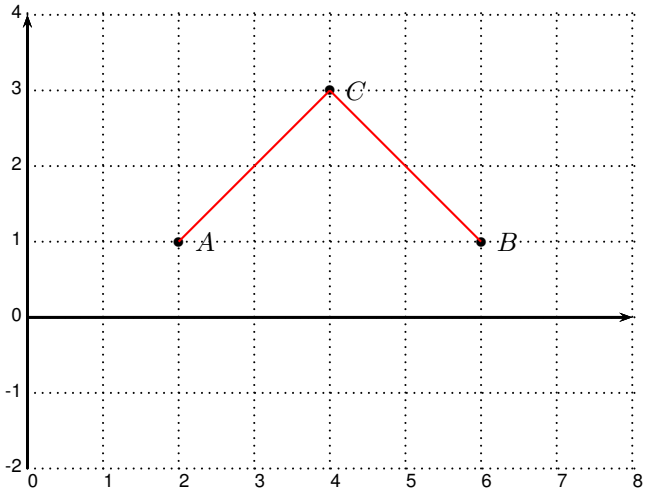
$$\begin{cases} x_C = \dfrac{(x_A + x_B)}{2} + \dfrac{(y_A - y_B)}{2} \\ y_C = \dfrac{(y_A + y_B)}{2} - \dfrac{(x_A - x_B)}{2} \end{cases}$$



2 triangles isocèles rectangles

2.3 Tracé

On trace alors les segments $[AC]$ et $[CA]$



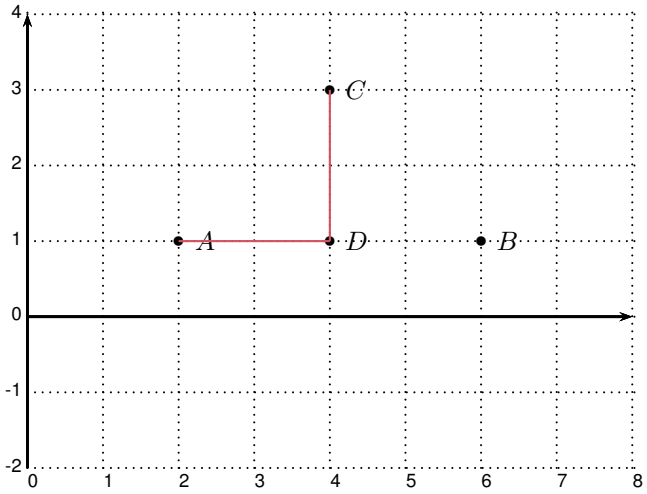
Fractale du dragon - $n = 1$

3 Rang n=2

Les coordonnées du point D sont données par les formules suivantes :

$$\begin{cases} x_D = \dfrac{(x_A + x_C)}{2} + \dfrac{(y_C - y_A)}{2} = 2 \\ y_D = \dfrac{(y_A + y_C)}{2} - \dfrac{(x_A - x_C)}{2} = 3 \end{cases}$$

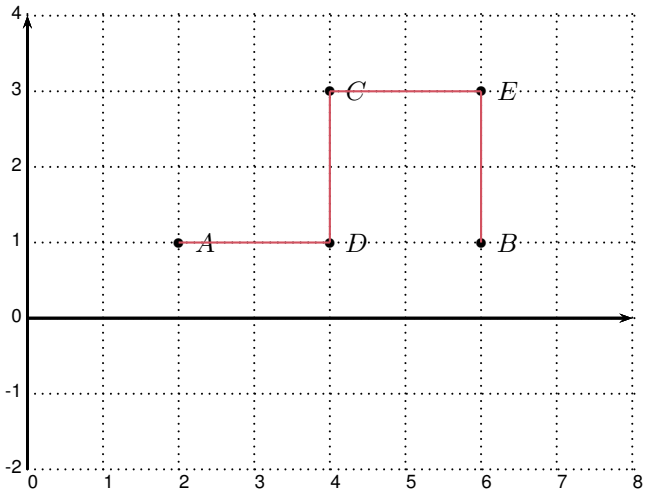
On trace alors les segments $[AD]$ et $[DC]$



Fractale du dragon - $n = 2$ - En construction

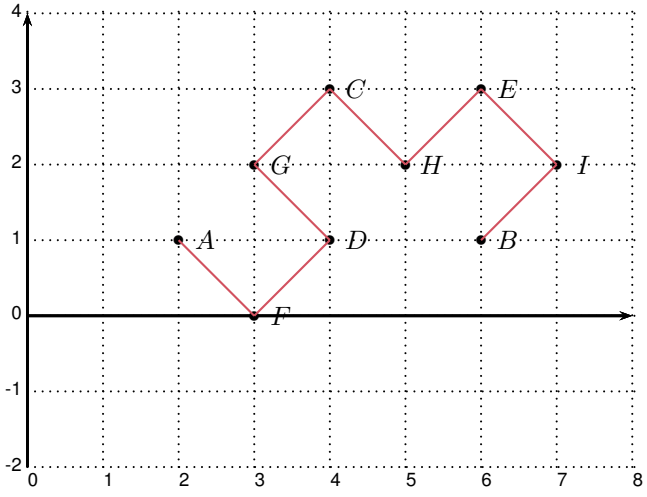
De la même façon, on obtient :

$$\begin{cases} x_E = \dfrac{(x_B + x_C)}{2} + \dfrac{(y_C - y_B)}{2} = 6 \\ y_E = \dfrac{(y_B + y_C)}{2} - \dfrac{(x_C - x_B)}{2} = 3 \end{cases}$$



Fractale du dragon - $n = 2$

4 Rang n=3



Fractale du dragon - $n = 3$

5 Programme avec Python et Tk

```
#!/usr/bin/python3

# Alexanot, le 16/01/2018 en Python 3
from tkinter import *

def dragon (n,x1,y1,x2,y2):
    if n==1:
        c.create_line (x1,y1,x2,y2,width=2,fill="red")
    else:
        x3=(x1+x2)/2 + (y2-y1)/2
        y3=(y1+y2)/2 + (x1-x2)/2
        dragon (n-1,x3,y3,x1,y1)
        dragon (n-1,x3,y3,x2,y2)

def effacer():
    global n
    c.delete (ALL)
    n=1

def animer():
    global n

```

```
    if n<=int (t.get()):
        c.delete(ALL)
        dragon (n,400,200,600,200)
        n=n+1
        f.after(1000,animer)

def quitter():
    global n
    f.destroy()
    n=1

n=1
f=Tk()
t = StringVar()
label = Label (f, text="␣Donner␣un␣entier␣")
label.pack()
entree = Entry (f, background='white', textvariable=t)
entree.pack()
c=Canvas(f, height=400,width=1000,bg='ivory')
c.pack()
e=Button(f, text="Effacer",command=effacer)
e.pack()
g=Button(f, text="Animer",command=animer)
g.pack()
b=Button(f, text="Quitter",command=quitter)
b.pack()
f.mainloop()
```

6 Programme avec Python et Turtle

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
# import the turtle module to use turtle graphics
import turtle

# make variables for the right and left containing 'r' and 'l'
r = 'r'
l = 'l'

# assign our first iteration a right so we can build off of it
old = r
new = old

# for inputs
iteration = int(input('Enter␣iteration:'))
length = 30
pencolor = "black"
bgcolor = "white"

# waiting for the fractal to be built
print ("Waiting␣...")

# set the number of times we have been creating
# the next iteration as the first
cycle = 1

# keep on generating the next iteration until desired iteration is reached
while cycle<iteration:
        # add a right to the end of the old iteration and save it to the new
        new = (old) + (r)
        # flip the old iteration around
        # (as in the first character becomes last)
        old = old[::-1]
```

```
        # cycling through each character in the flipped old iteration:
        for char in range(0, len(old)):
                # if the character is a right:
                if old[char] == r:
                        # change it to a left
                        old = (old[:char])+ (l) + (old[char + 1:])
                # otherwise, if it's a left:
                elif old[char] == l:
                        #change it to a right
                        old = (old[:char]) + (r) + (old[char + 1:])
        # add the modified old to the new iteration
        new = (new) + (old)

        # save the new iteration to old as well for use next cycle
        old = new

        # advance cycle variable to keep track of the number of times
        # it's been done
        cycle = cycle + 1


printans = "y"

# for not show the turtle icon when drawing
turtle.ht()
turtle.speed(0)
turtle.color(pencolor)
turtle.bgcolor(bgcolor)
turtle.forward(length)

# cycling through all the characters in the iteration
for char in range(0, len(new)):
        # if the character is a right:
        if new[char] == (r):
                turtle.right(90)
                turtle.forward(length)
        # otherwise, if the character is a left:
        elif new[char] == (l):
                turtle.left(90)
                turtle.forward(length)

input()
```