Python et la tortue : découvrir la programmation en dessinant

Deux activités de découverte de la programmation utilisant la bibliothèque turtle de python pour dessiner sur l'écran sont présentées ici. La première consiste à tracer des courbes de Bézier, et la seconde une courbe fractale. Bien qu'elles ne nécessitent pas de très nombreuses lignes de code, il ne s'agit cependant pas d'activités de niveau débutant, car elles ont pour but d'illustrer la notion de récursivité.

Pascal Lafourcade

Maître de conférences à l'IUT d'informatique de l'Université Clermont Auvergne et membre du Laboratoire d'Informatique, Modélisation et Optimisation des Systèmes. Il est spécialiste en sécurité informatique et cryptographie.

Malika More

Maîtresse de conférences à l'IUT d'informatique de l'Université Clermont Auvergne et membre du Laboratoire d'Informatique, Modélisation et Optimisation des Systèmes. Elle est responsable du groupe Informatique Sans Ordinateur de l'IREM de Clermont-Ferrand.

LA BIBLIOTHÈQUE TURTLE

Python possède une bibliothèque à vocation pédagogique appelée « turtle ». Elle offre la possibilité de créer des programmes à l'aide d'instructions proches du LOGO, un célèbre langage de programmation éducatif inventé par Wally Feurzig et Seymout Papert en 1967 au Massachusetts Institute of Technology.

La documentation de la bibliothèque se trouve ici :
https://docs.python.org/fr/3/library/turtle.html

Comme toute bibliothèque python, les lignes import turtle ou from turtle import * en début de programme permettent de l'utiliser. Elle fait appel au module tkinter pour sa partie graphique.

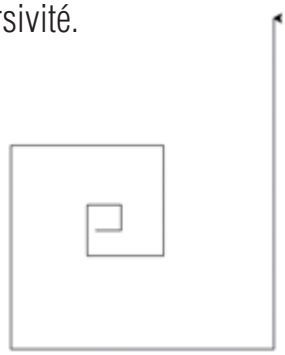
Les commandes ont pour objectif de contrôler sur l'écran un objet appelé « tortue », qui permet de dessiner avec un ensemble de commandes simples dans une fenêtre graphique. Le nom vient de la « tortue logo », un véritable robot piloté à l'aide du langage logo, utilisé dans certaines écoles dans les années 1970 et 1980. Il est à noter que le rôle de la tortue était aussi parfois joué par un enfant (ou un adulte) à quatre pattes, dans une démarche annonciatrice du courant de l'informatique débranchée, formalisé au début des années 2000 par Tim Bell et ses collègues en Nouvelle-Zélande. https://csunplugged.org/fr/

Pour déplacer la tortue et dessiner avec, nous pouvons utiliser les commandes suivantes :

- forward() permet d'avancer d'une distance donnée
- backward() permet de reculer d'une distance donnée
- right() permet de tourner à droite d'un angle donné (en degrés)
- left() permet de tourner à gauche d'un angle donné (en degrés)
- goto() permet d'aller à une position définie par deux coordonnées (à noter que le point (0,0) est au centre de la fenêtre)
- pendown() met le stylo en position d'écriture
- penup() lève le stylo
- clear() efface tout ce qui est dessiné

Il existe beaucoup d'autres commandes, mais les huit qui sont listées ci-dessus permettent de réaliser de nombreuses activités pour découvrir divers aspects de la programmation.

Voici un exemple permettant de tracer une spirale rectangulaire. Bien qu'il ne contienne que cinq lignes de code, il contient deux notions importantes pour les débutants : la boucle for et la réutilisation des variables.



```
from turtle import *

longueur = 30
for i in range(10):
    forward(longueur)
    left(90)
    longueur = longueur+10*i
```

COURBES DE BÉZIER

Les courbes de Bézier ont été inventées en 1962 par Pierre Bézier, un ingénieur travaillant chez Renault. Elles sont aujourd'hui à la base des images vectorielles, et très utilisées en CAO, en synthèse d'image et pour le rendu des polices de caractères.

La définition mathématique des courbes de Bézier fait appel à des polynômes. Les plus utilisées sont celles qui correspondent à des polynômes de degré 3, mais ce ne sont pas celles présentées ici, par souci de simplicité.

Les courbes de Bézier présentées ici sont dites « *quadratiques* » (elles sont définies par des polynômes de degré 2) et ont été proposées par Paul de Faget de Casteljau, un ingénieur travaillant chez Citroën). Elles utilisent trois paramètres : le point de départ A, le point d'arrivée C et un point de contrôle B. Pour comprendre le fonctionnement d'une telle courbe, il faut imaginer que notre segment [A;C] est un fil de métal souple, qui est attiré par l'aimant qu'est le point de contrôle B. Plus une partie du fil se trouve près de l'aimant, plus elle est attirée et se déforme.

La courbe de Bézier de point de départ A, de point d'arrivée C et de point de contrôle B, est la limite d'une suite de lignes brisées, obtenues à l'aide d'un algorithme également inventé par Paul de Casteljau.

La ligne initiale est formée des segments [AB] et [BC].

À l'étape suivante, les trois points suivants sont construits :

- D milieu de [AB]
- E milieu de [BC]

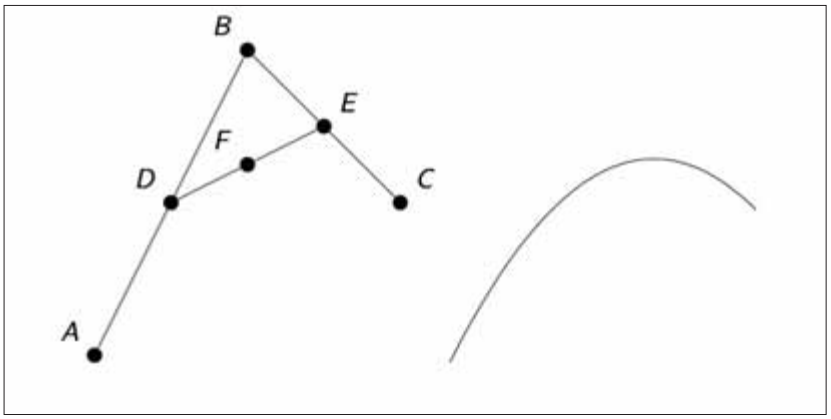


Figure 1

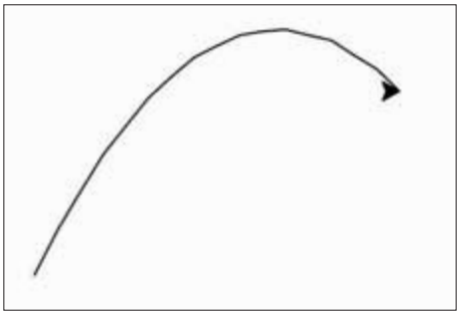


Figure 2

• F milieu de [DE]

Il faut noter que D et E sont des points auxiliaires, tandis que F est un point de la courbe finale.

Ensuite, il faut recommencer en remplaçant les points A, B, C par les deux triplets de points A, D, F et F, E, C. Après plusieurs itérations, le résultat est une courbe lisse, passant par A et C et tangente à [AB] et [BC]. Le résultat est déjà reconnaissable après 3 ou 4 itérations. **Figure 1**

Pour implémenter cet algorithme, une solution est de commencer par écrire une fonction nommée segment, qui trace un segment à partir des coordonnées de deux points.

```
from turtle import *

def segment(x1,y1,x2,y2) :
    up()
    goto(x1,y1)
    down()
    goto(x2,y2)
    up()
    return
```

Ensuite une autre fonction utile est le calcul des coordonnées du milieu de deux points à partir de deux points.

```
def milieu(x1,y1,x2,y2) :
    return ((x1+x2)/2.0,(y1+y2)/2.0)
```

À partir de ces deux fonctions, une des manières naturelles de coder cet algorithme est de réaliser une fonction récursive pour tracer la courbe de Bézier.

```
def bezier(x1,y1,x2,y2,x3,y3,n) :
    if n==0 :
        segment(x1,y1,x2,y2)
        segment(x2,y2,x3,y3)
    else :
        (a,b)=milieu(x1,y1,x2,y2)
        (c,d)=milieu(x2,y2,x3,y3)
        (e,f)=milieu(a,b,c,d)
        bezier(x1,y1,a,b,e,f,n-1)
        bezier(e,f,c,d,x3,y3,n-1)
```

Le résultat de la fonction bezier pour les points trois-points (0,0), (100,200) et (200,100) et une profondeur de 3 donne le graphique suivant (en appelant bezier(0,0,100,200, 200,100,3)) : **Figure 2**

L'algorithme de Casteljau est ici présenté pour construire des courbes de Bézier quadratiques, mais la construction se généralise assez facilement pour les courbes de Bézier de degré supérieur.

FLOCON DE VON KOCH

Le flocon de von Koch est une courbe fractale simple à tracer. Il s'agit d'un d'un polygone « creux » (non convexe). La figure de départ est un triangle équilatéral, et l'algorithme de construction s'applique à chaque côté du triangle. La construction est présentée étape par étape : pour passer d'une étape à la suivante, il faut diviser chaque segment en trois segments égaux. Ensuite, il faut remplacer le segment du milieu par les deux côtés d'un triangle équilatéral dont la base est ce segment du milieu, et orienté vers l'extérieur. À l'étape suivante, ce processus est réitéré sur les quatre nouveaux segments obtenus.

Cette construction correspond naturellement à un algorithme récursif, comme celui écrit ci-dessous.

```
from turtle import *

def Trait(n, L) :
    # Etat initial : plume baissée, direction trait, à gauche
    # Etat final :    plume baissée, direction trait, à droite
    if n == 1 :   forward(L)
    else :
        Trait(n-1, L / 3.0)
        left(60)
        Trait(n-1, L / 3.0)
        right(120)
        Trait(n-1, L / 3.0)
        left(60)
        Trait(n-1, L / 3.0)

N = int(input("Nombre de niveaux (entre 1 et 6) : "))
if (type(N) != int) or (N < 1) or (N > 6) :
    print ("Le nombre de niveaux ne correspond pas a la demande")
else :
    Long = 550    # Longueur du trait
    up()
    goto(-300, -180)
    down()
    width(2) # épaisseur du stylo
    left(60)
    for k in range(3) :
        Trait(N, Long)
```

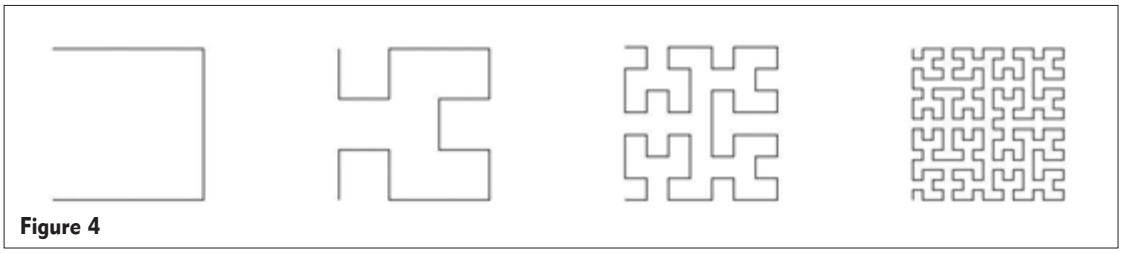


Figure 4

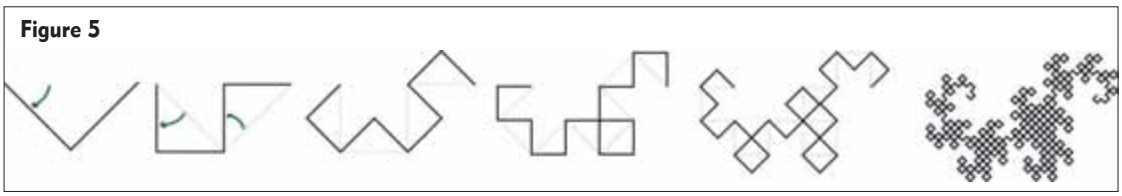
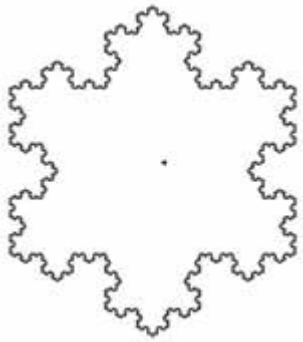


Figure 5



FLOCON DE VON KOCH
Figure 3

```
    right(120)
up()
goto(0, 0)
```

En théorie, la construction devrait être répétée indéfiniment pour obtenir une courbe fractale, car par définition, il s'agit de la courbe « limite », obtenue après une infinité d'étapes. Concrètement, on fixe préalablement la profondeur des appels récursifs. Dans le programme ci-dessous elle est limitée à 6, car d'une part le temps d'exécution est déjà long, et d'autre part, il n'y aurait plus beaucoup de différence visuelle en ajoutant des étapes. **Figure 3**
Il existe des variantes du flocon de von Koch, obtenues en modifiant les valeurs des angles du triangle ou bien la figure géométrique de base, et même en 2D (surfaces de von Koch).

POUR ALLER PLUS LOIN

Voici deux exemples supplémentaires de courbes fractales, la courbe de Hilbert et la courbe du dragon, dont il est possible de tracer des approximations à l'aide de la bibliothèque turtle.

Courbe de Hilbert

Cette courbe, inventée par le mathématicien allemand David Hilbert en 1891, est une ligne brisée, obtenue à partir de segments de droites orthogonaux. Le motif de base est composé de trois côtés d'un carré. À chaque étape, la longueur des côtés est divisée par 3. Chaque « coin » d'un motif de l'étape précédente est remplacé par le nouveau motif, selon la même orientation, et chaque « extrémité » est aussi remplacée par le nouveau motif, mais tourné d'un quart de tour, chacun dans un sens, de façon que les ouvertures soient vers l'extérieur. Il ne reste plus qu'à relier de proche en proche les extrémités des motifs pour obtenir la courbe suivante : **Figure 4**

La courbe fractale obtenue après une infinité d'itérations possède la propriété de passer par chacun des points du carré bâti à partir du motif initial.

Courbe du dragon

La courbe du dragon a été inventée par J.E. Heighway. Cette courbe fractale ne se recoupe jamais. En 1967, Martin Gardner l'a présentée dans sa chronique de jeux mathématiques du Scientific American, car sa construction est simple. À chaque étape, la courbe obtenue est composée de segments de droites qui suivent à angle droit. Au départ, il n'y a qu'un seul segment. Pour passer d'une étape à la suivante, en suivant la courbe, il s'agit de remplacer chaque segment rencontré par deux segments à angle droit en effectuant une rotation de 45° alternativement à droite puis à gauche. **Figure 5**

CONCLUSION

Nous espérons vous avoir convaincu de l'intérêt de la bibliothèque turtle de python pour la découverte de la programmation par le dessin, y compris pour des notions avancées comme la récursivité. Les codes Python pour les courbes de Hilbert et du dragon sont laissés en exercices au lecteur, pour qu'il puisse vérifier s'il a compris le fonctionnement de cette bibliothèque et la récursivité.