

# **Générateur de L-Systemes**

**Baptiste Rivet**

**Projet D'ISN 2016-2017 – Lycée Jean-Baptiste Say – M. Risolo**

## Qu'est ce qu'un L-Système ?

Un L-Système est un système d'écriture servant à modéliser le comportement d'une fractale. Il est écrit sous forme d'une chaîne de caractères, chacun représentant une action.

Un L-Système se compose de quatre éléments :

- Un **alphabet**, qui est l'ensemble des caractères variables que l'on peut utiliser dans le L-système.
  - Un caractère est un symbole qui est associé à une action pour tracer le L-système
- Des **constantes** qui sont des symboles prédéfinis et communs à tous les L-systèmes
- Un **axiome**, qui est la chaîne du caractère initiale du L-Système
- Un ensemble de **règles**, qui définissent l'évolution du L-Système

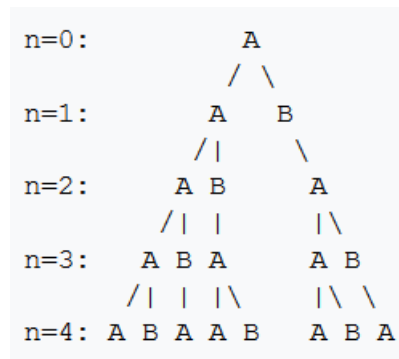
Prenons comme exemple le L-Système suivant

```
{  
  Alphabet : {A, B}  
  Constantes : {}  
  Axiome : A  
  Règles : {(A -> AB), (B -> A)}  
}
```

A l'étape  $n = 0$ , la chaîne de caractère du L-Système est l'axiome (A).

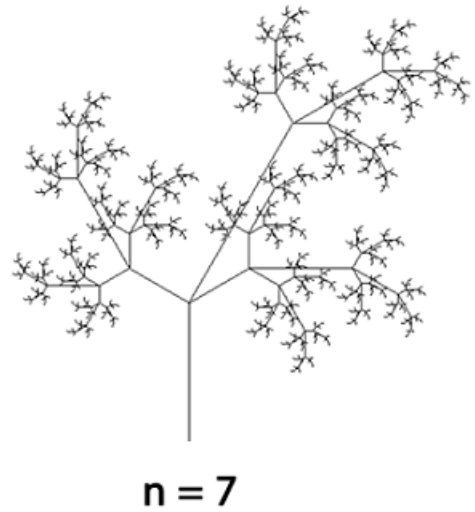
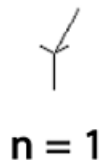
A l'étape  $n = 1$ , on applique la règle (A -> AB) au A est le L-Système est (AB).

Pour les étapes suivantes on répète l'opération en appliquant les règles qu'il faut et on obtient, pour  $n = 3$ , (ABA)



Ces caractères font effectuer des actions (que l'on a associé à chacun d'entre eux) à une tortue sur un canvas. Et c'est le dessin réalisé par la tortue qui fait la fractale.

Exemple d'une fractale réalisée par la tortue :



## But du programme

Le programme a pour but d'aider les utilisateurs de celui-ci à construire simplement des L-Systèmes. En spécifiant les caractéristiques d'un L-Système citées précédemment, il pourrait tracer un L-Système et voir son évolution en fonction du niveau de récurrence indiqué.

## Le langage LS

Afin que les personnes puissent programmer leurs L-Systèmes dans une logique propre à ceux-ci, nous avons créé un "langage de programmation" dans lequel les utilisateurs pourraient déclarer séparément chaque variable d'un L-Système. Nous avons décidé de les faire utiliser ce langage à la place du Python car la conception des objets n'est pas tout à fait en accord avec la manière de penser pour créer des L-Système. Par exemple, pour écrire une chaîne d'instruction du type "R1 = A+B[-AB]+A", il aurait fallu écrire "R1 = [A, PLUS, B, OUVERT, MOINS, A, B, FERME, PLUS, A]". Voici les instructions que l'on trouve dans le LS :

- Caractere(A = FORWARD(10)) pour créer le caractère A qui a comme effet de faire avancer la tortue d'un pas de 10 pixels.
- Regle(R1 = A->A+B) pour créer la règle R1, qui a comme effet de changer A en A+B.
- Lsystem(Pythagore)
  - {
    - Alphabet = [A, B]
    - Axiome = ABA
    - Regles = [R1, R2]
    - Pas = 10
    - Angle = 25
  - }

pour créer le L-Système nommé Pythagore, avec A et B dans son alphabet, ABA comme axiome, qui suit les règles R1 et R2, avec un pas de 10 pixels et des angles de 30 degrés.
- Generer(Pythagore)

```

{
    N = 10
    Position = GAUCHE, 100
    Position = BAS, 100
}

```

pour afficher le L-Système Pythagore au niveau n = 10, avec l'origine à 100 pixels de la gauche de l'écran et du bas de l'écran.

## Fonctionnement général

Le programme est composé de 5 parties principales.

- La première partie est la fenêtre principale (**programme.py**), c'est ce fichier que l'utilisateur lancera pour lancer le programme. Il fait apparaître un éditeur de texte, qui permettra à l'utilisateur d'entrer son code de L-Système et de l'enregistrer dans un fichier texte. Il pourra ensuite le faire tracer grâce à une fenêtre Turtle.
- Le compilateur (**compilateur.py**) est la partie du programme qui se chargera de transformer la saisie de l'utilisateur en code python qui pourra être exécuté ensuite.
- Le code compilé (**compile.py**) est le fichier qui résulte de la compilation du texte écrit avec l'éditeur par le compilateur, c'est ce fichier qui, lancé, tracera le L-Système dans une fenêtre Turtle.
- La bibliothèque (**Caractere.py**) est le fichier qui contient les classes et les fonctions qui définissent un L-Système, ainsi que la manière dont il doit être tracé. C'est ces fonctions qui sont utilisées dans le fichier compilé.
- Les fichiers transitions sont des morceaux de codes qui servent uniquement à transmettre des informations d'une partie du programme à une autre, ou bien à exécuter un code. Ils sont au nombre de 3 (**nomFichier.py**, **batch.bat**, **batch2.bat**).

L'utilisateur commence par écrire son code dans la zone de texte de la fenêtre principale. Ensuite, il l'enregistre dans un fichier texte. Lorsqu'il appuie sur le bouton "compiler", le programme écrit le nom du fichier à compiler dans "nomFichier.py". Il lance ensuite le programme "batch.bat" qui lance le compilateur. Le compilateur ouvre donc le fichier indiqué dans nomFichier puis compile le code et l'écrit dans "compile.py". L'utilisateur appuie ensuite sur le bouton "executer", qui exécute "batch2.bat", et qui lance le fichier compilé et donc la fenêtre Turtle, qui trace la fractale.

Nous avons codé ce programme en Python, avec les bibliothèques Turtle pour la tortue qui dessine le L-Système, Tkinter pour la fenêtre graphique, et os pour pouvoir exécuter des fichiers. Nous avons aussi utilisé le langage batch, qui effectue des actions dans Windows, pour lancer des programmes qui sont dispersés dans l'ordinateur.

## La fenêtre principale

La fenêtre principale est la seule interface avec laquelle l'utilisateur aura à interagir. C'est avec celle-ci qu'il écrira son code, qu'il le compilera et l'exécutera.










La fenêtre est composée en 4 parties :

La première partie est la zone de texte, c'est dans celle-ci que l'on rentre un code de L-Systeme. Cette partie a été réalisée avec le widget ScrolledText de Tkinter, qui permet d'avoir une zone de texte avec un barre de défilement pour naviguer à travers le texte.

La deuxième partie, sur la gauche de l'écran, est la barre de macros, si l'utilisateur clique sur le bouton associé à une fonction, le programme insèrera le code minimum de la fonction, que l'utilisateur n'aura plus qu'à remplir. Par exemple, en appuyant sur le bouton "Generer()", le texte suivant sera inséré dans la zone de texte :

```
Generer()
{
    N =
}
```

En haut de la zone de texte se trouve la barre des boutons. Sur cette barre se situent des boutons sur lesquels l'utilisateur pourra appuyer afin de profiter des actions de ceux-ci. Les boutons disponibles sont :

-  Nouveau : permet de créer un nouveau fichier texte.
-  Ouvrir : ouvre un fichier existant dans l'éditeur de texte.
-  Enregistrer : enregistre le fichier actuel.
-  Enregistrer sous : enregistrer le fichier actuel sous.
-  Annuler : annule la dernière action.
-  Rétablir : rétablit la dernière action annulée.
-  Compiler : lance la compilation du programme.
-  Executer : execute le programme compilé.
-  Compiler et lancer : lance puis execute le programme.

Chaque bouton de la fenêtre lance une fonction du programme, qui effectue l'action indiquée.

Enfin, tout en haut de la fenêtre, il y a la barre de menu, qui permet d'accéder aux mêmes fonction qu'avec les boutons, avec certaines options en plus.

L'interface a été codée grâce à la bibliothèque Tkinter, et ses nombreux widgets. On retrouve notamment ScrolledText pour la zone de texte, Button pour les boutons, Menu pour la barre de menu et filedialog pour les fenêtres d'ouverture d'un fichier. Enfin, le tout est mis en page grâce à des PanedWindow.

## Le compilateur

Le compilateur commence par ouvrir le fichier `nomFichier.py` qui lui donne le nom du fichier à compiler. Il ouvre ensuite le fichier à compiler et stocke le texte contenu dedans. Le programme examine ensuite ce fichier ligne par ligne, il regarde à quelle instruction python la ligne correspond en cherchant un mot-clé dans la ligne, extirpe les paramètres de la ligne de code et écrit dans le fichier compilé l'instruction correspondante en python.

Voici un exemple : Le programme doit compiler `"Caractere(A = FORWARD(10))"`. Avec la fonction `find`, le programme remarque la présence de `"Caractere"` dans cette ligne. Il supprime donc les informations superflues pour pouvoir récupérer les paramètres de la fonction : `'A'`, `'FORWARD'` et `10`. Le compilateur écrira donc `"A = Caractere('A', FORWARD, 10, 0)"` dans le fichier compilé.

Une fois arrivé à la fin du fichier, il écrira les instructions manquantes pour faire fonctionner le programme, c'est à dire `"from Caractere import *"` et `"mainloop()"`, avant de quitter.

## La bibliothèque Caractère

La bibliothèque Caractère est en quelques sortes le coeur du programme, c'est ici que sont déclarées les fonctions et les classes utilisées par le fichier compilé pour tracer le L-Système. Voici les différents objets que cette bibliothèque contient :

- `Caractere` est une classe qui est utilisée pour définir les caractères utilisés dans le L-Système. A chaque caractère est associé un symbole qui sera utilisé dans la chaîne de caractère du L-Système (ex : `'A'`). Ensuite, il faut lui associer une action à faire effectuer à la tortue lorsque ce caractère est lu. Il y a 8 actions qui peuvent être associées à un `Caractere` :
  - `NUL` : Aucune action ne se passe
  - `FORWARD` : Avance d'un pas
  - `BACKWARD` : Recule d'un pas
  - `LEFT` : Tourne à gauche d'un pas angulaire
  - `RIGHT` : Tourne à droite d'un pas angulaire
  - `ROLL` : Inverse la direction
  - `SAVE` : Sauvegarde la position et l'orientation de la tortue (à noter que sauvegarder une position n'efface pas la précédente)
  - `LOAD` : Revient à la dernière position sauvegardéeEnsuite, on peut spécifier une option pour l'action, celle-ci sera soit le pas pour `FORWARD` et `BACKWARD`, soit l'angle pour `LEFT` et `RIGHT`. On peut aussi décider de définir une fonction (expliqué au point suivant) comme paramètre pour les actions. Si l'on ne spécifie ni une option ni une fonction, l'ordre suivra la valeur indiquée lors de la création du L-Système.  
Exemple : `A = Caractere("A", FORWARD, 10, 0)` définit `"A"`, qui aura pour action de faire avancer la tortue de 10 pixels.
- `Fonction` est une classe qui définit une fonction affine. Cette fonction, précisée en argument pour un caractère, permet de faire varier une distance ou un angle en fonction du niveau de récursivité du L-Système. Cette classe prend pour argument un coefficient directeur et une ordonnée à l'origine.

Par exemple :  $f1 = Fonction(3, 2)$  définit la fonction  $f1(n) = 3n+2$ .

Si l'on écrit  $B = Caractere("B", FORWARD, 0, f1)$ , B aura pour effet de faire avancer la tortue d'un pas de 5 pour  $n = 1$ , et d'un pas de 8 pour  $n = 2$ .

- Regle est une classe qui définit les règles d'un L-système. Elle prend comme arguments début, qui est le caractère qu'elle transforme, et fin, qui est un tableau de Caractere, le résultat de la transformation du caractère du début.

Exemple :  $R1 = Regle(A, [A, B])$  est une règle qui transforme A en AB.

- La classe Lsysteme est la classe principale du programme, celle qui définit un L-Système. Elle a comme attributs :
  - alphabet : un tableau de caractères qui définit l'alphabet du L-Système.
  - axiome : le caractère qui est l'axiome du L-Système
  - regles : un tableau de regles qui sont celles qui s'appliqueront au L-Système
  - pas : le pas par défaut du L-Système
  - angle : l'angle par défaut du L-Système (pour pas et angle, ils sont utilisés lorsque il n'y a ni option, ni fonction déclarées dans le caractère.

Ses méthodes sont :

- nouveauCaractere : Ajoute un nouveau caractère à l'alphabet.
- nouvelAxiome : Remplace l'axiome precedent par un nouveau.
- nouvelleRegle : Ajoute une regles a celles deja indiquees.
- nouveauPas : Definit un nouveau pas pour la figure.
- nouvelAngle : Definit un nouvel angle pour la figure.
- generer : Cree la chaine de caractere du L-systeme a la n-ieme etape.
- produire : Affiche la figure produite par le L-systeme avec generer.
- position : change la position d'orgine du tracé du L-système.

Dans ce fichier sont aussi définies les constantes, qui sont des Caractere, mais qui sont utilisables sur tout les L-Systèmes, il utilisent les valeurs d'angles et de distance par défaut du L-Système. Elles sont au nombre de 5 :

- + (PLUS) : équivalent de LEFT
- - (MOINS) : équivalent de RIGHT
- | (BARRE) : équivalent de ROLL
- [ (OUVERT) : équivalent de SAVE
- ] (FERME) : équivalent de LOAD

## Les fichiers Batch

Les fichiers Batch n'ont pour seule utilité que d'exécuter une partie du programme sans que l'utilisateur aie à le faire manuellement. Le fichier batch.bat lance le compilateur depuis le programme principal, tandis que batch2.bat lance le fichier compilé.

## Essai du programme

Dans cette partie, nous allons essayer de faire marcher le programme. Nous allons le faire avec le triangle de Sierpinski. Nous commençons par écrire le code suivant dans l'interface :

```
Caractere(F = FORWARD(0))
Caractere(G = FORWARD(0))
```

```
Regle(R1 = F->F-G+F+G-F)
Regle(R2 = G->GG)
```

```
LSysteme(Sierpinski)
{
    Alphabet = [F, G]
    Axiome = F-G-G
    Regles = [R1, R2]
    Pas = 5
    Angle = 120
}
Generer(Sierpinski)
{
    N = 6
}
```

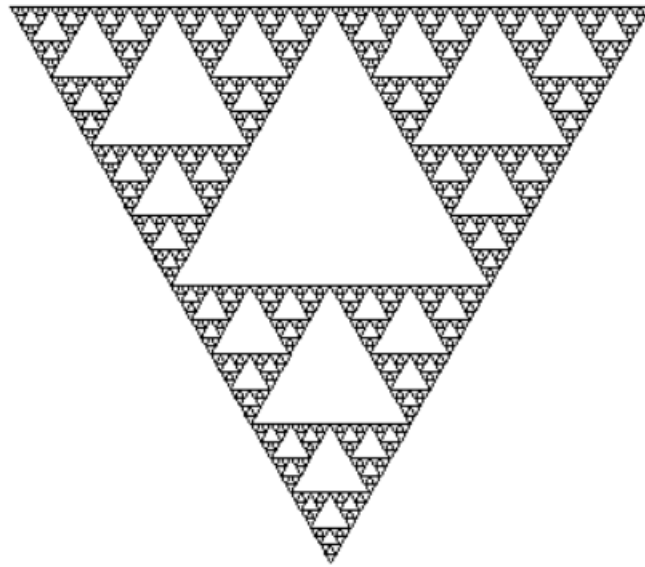
Le code compile et se transforme en ceci :

```
from Caractere import *

F = Caractere("F", FORWARD, 0, 0)
G = Caractere("G", FORWARD, 0, 0)
R1 = Regle(F, [F, MOINS, G, PLUS, F, PLUS, G, MOINS, F])
R2 = Regle(G, [G, G])
Sierpinski = LSysteme([F,G], [F, MOINS, G, MOINS, G], [R1,R2], 5, 120)
Sierpinski.generer(6)
Sierpinski.produire()
mainloop()
```

Enfin, une fenêtre Turtle se lance et nous obtenons la figure voulue :





## Ce que l'on peut encore faire

Il y a différentes choses que l'on aurait aimé rajouter au programme mais que nous n'avons pas eu le temps de faire. Il y a tout d'abord les L-Systèmes stochastiques, qui sont des L-Systèmes avec plusieurs règles pour un seul caractère, chacune ayant une probabilité d'être choisie. Il y a aussi des L-Systèmes avec des règles dépendant du contexte dans lequel le paramètre se trouve. Par exemple, un B se transformera en A si et seulement si il est entouré par 2 autres B. Finalement, il y a les L-Systèmes paramétriques, où chaque caractère possède des paramètres qui évoluent. Par exemple  $A(x, y)$  devient  $A(x-1, y+1)$  si  $x = y$ .

## Le code du programme

### programme.py

```
from tkinter import *
from tkinter.messagebox import *
from tkinter.scrolledtext import *
from tkinter.filedialog import *
from time import *
from os import system

global fichierActuel
fichierActuel = ""

#//////////////////FONCTIONS BARRE D'OUTILS//////////////////

def new():
    """
```

```

        Crée un nouveau fichier texte. Sauvegarde d'abord le fichier ouvert.
    """
    global fichierActuel
    if fichierActuel == "":
        if askyesno("Fichier non sauvegardé", "Voulez-vous sauvegarder le fichier actuel ?"):
            enregistrerSous()
            realNew()
        else:
            realNew()
    else:
        fichier = open(fichierActuel, "r")
        contenu = fichier.read()
        mot = str(entree.get("0.0", "end"))
        if contenu == mot:
            fichier.close()
            realNew()
        else:
            fichier.close()
            if askyesno("Fichier non sauvegardé", "Voulez-vous sauvegarder le fichier actuel ?"):
                enregistrer()
                realNew()
            else:
                realNew()

def realNew():
    """
        Appelée par new() pour effacer le texte à l'écran
    """
    global fichierActuel
    entree.delete("0.0", "end")
    fichierActuel = ""

def undo():
    """
        Annule la dernière action.
    """
    entree.edit_undo()

def redo():
    """
        Annule les actions annulées.
    """
    entree.edit_redo()

def enregistrer():
    """
        Enregistre le code dans un fichier.
    """
    global fichierActuel
    mot = str(entree.get("0.0", "end"))
    if fichierActuel == "":
        enregistrerSous()

```

```

elif fichierActuel == ".txt":
    return 1
else:
    fichier = open(fichierActuel, "w")
    fichier.write(mot)
    fichier.write("\n")
    fichier.close()

def enregistrerSous():
    """
        Enregistre le code sous dans un fichier.
    """
    global fichierActuel
    fichierActuel = str(asksaveasfilename(title="Enregistrer sous", filetypes=[("txt files", ".txt"),("all
files", ".*)"])))
    txtFind = fichierActuel.find(".txt")
    if txtFind == -1:
        fichierActuel += ".txt"
    enregistrer()

def ouvrir():
    """
        Ouvre un fichier et l'affiche.
    """
    global fichierActuel
    filepath = askopenfilename(title="Ouvrir un fichier", filetypes=[("txt files", ".txt"),("all files", ".*)"]])
    if filepath == "":
        return 1
    fichier = open(filepath, "r")
    contenu = fichier.read()
    fichier.close()
    entree.delete("0.0", "end")
    entree.insert("insert", contenu)
    fichierActuel = str(filepath)

def buildAndRun():
    build()
    sleep(1)
    run()

def build():
    fichier2 = open("nomFichier.py", "w")
    fichier2.write('def nomFichier():\n\treturn "')
    fichier2.write(fichierActuel)
    fichier2.write('"')
    fichier2.close()
    system("batch.bat")

def run():
    enregistrer()
    system("batch2.bat")

```

```

def addLSysteme():
    entree.insert("insert", "LSysteme()\n{\n\tAlphabet = []\n\tAxiome = \n\tRegles = []\n\tPas = \n\tAngle = \n}")

def addCaractere():
    entree.insert("insert", "Caractere( = ())")

def addGenerer():
    entree.insert("insert", "Generer()\n{\n\tN = \n}")

def addRegle():
    entree.insert("insert", "Regle( = ->)")

#####INTERFACE#####

fenetre = Tk() #Fenetre principale (la fenetre Windows)

#---Def Icones Barre D'Outils---
photoRun = PhotoImage(file="run.gif") #Executer
photoBuild = PhotoImage(file="build.gif") #Compiler
photoBuildAndRun = PhotoImage(file="buildAndRun.gif") #Compiler et Executer
photoFileNew = PhotoImage(file="filenew.gif") #Nouveau
photoFileOpen = PhotoImage(file="fileopen.gif") #Ouvrir
photoFileSave = PhotoImage(file="filesave.gif") #Enregistrer
photoFileSaveAll = PhotoImage(file="filesaveall.gif") #Enregistrer Sous
photoUndo = PhotoImage(file="undo.gif") #Annuler (Ctrl+Z)
photoRedo = PhotoImage(file="redo.gif") #Revenir (Ctrl+Y)
#-----

#---Barre D'Outils
menubar = Menu(fenetre)

menu1 = Menu(menubar, tearoff=0)
menu1.add_command(label="Nouveau", command=new)
menu1.add_command(label="Ouvrir", command=ouvrir)
menu1.add_command(label="Enregistrer", command=enregistrer)
menu1.add_command(label="Enregistrer sous", command=enregistrerSous)
menu1.add_command(label="Imprimer")
menu1.add_command(label="Quitter")

menubar.add_cascade(label="Fichier", menu=menu1)

menu2 = Menu(menubar, tearoff=0)
menu2.add_command(label="Annuler", command=undo)
menu2.add_command(label="Rétablir", command=redo)
menubar.add_cascade(label="Édition", menu=menu2)

fenetre.config(menu=menubar)

PO = PanedWindow(fenetre, orient=VERTICAL, height=720, width=1280)

```

```

P0.pack(side=LEFT, expand=Y, fill=BOTH)

P1 = PanedWindow(P0, orient=HORIZONTAL)
P2 = PanedWindow(P1, orient=VERTICAL)

entree = ScrolledText(P2, height=33)

#---

P2.add(entree)
P2.add(Label(P2, text="Log", background="blue", anchor="center"))

P11 = PanedWindow(P1, orient=VERTICAL)

P11.add(Button(P11, text="LSysteme()", width=20, command=addLSysteme))
P11.add(Button(P11, text="Generer()", width=20, command=addGenerer))
P11.add(Button(P11, text="Caractere()", width=20, command=addCaractere))
P11.add(Button(P11, text="Regle()", width=20, command=addRegle))
P11.add(Frame(P11))

P1.add(P11)
P1.add(P2)

P01 = PanedWindow(fenetre, orient=HORIZONTAL)
P011 = PanedWindow(fenetre, orient=HORIZONTAL, borderwidth=2, relief=RIDGE)
P012 = PanedWindow(fenetre, orient=HORIZONTAL, borderwidth=2, relief=RIDGE)
P013 = PanedWindow(fenetre, orient=HORIZONTAL, borderwidth=2, relief=RIDGE)

P011.add(Button(P011, image=photoFileNew, command=new))
P011.add(Button(P011, image=photoFileOpen, command=ouvrir))
P011.add(Button(P011, image=photoFileSave, command=enregistrer))
P011.add(Button(P011, image=photoFileSaveAll, command=enregistrerSous))

P01.add(P011)

P012.add(Button(P012, image=photoUndo, command=undo))
P012.add(Button(P012, image=photoRedo, command=redo))

P01.add(P012)

P013.add(Button(P013, image=photoBuild, command=build))
P013.add(Button(P013, image=photoRun, command=run))
P013.add(Button(P013, image=photoBuildAndRun, command=buildAndRun))

P01.add(P013)

P01.add(Frame(P01))

P0.add(P01)
P0.add(P1)

P0.add(P1)

```

```
P0.pack()
```

```
fenetre.mainloop()
```

## **compileur.py**

```
from nomFichier import *
```

```
mon_fichier = open(nomFichier(), "r")
```

```
compiler = open("compile.py", "w")
```

```
contenu = mon_fichier.read()
```

```
strbuff = ""
```

```
removeBuff = ""
```

```
boolL = False
```

```
boolGenerer = False
```

```
compteurPos = 0
```

```
compiler.write("from Caractere import *\n\n")
```

```
while contenu != "":
```

```
    boolN = False
```

```
    for element in contenu:
```

```
        if element == '\n':
```

```
            boolN = True
```

```
        if boolN == False:
```

```
            strbuff += element
```

```
    # print(strbuff)
```

```
    removeBuff = str(strbuff) + '\n'
```

```
    a1 = strbuff.find("Caractere")
```

```
    a2 = strbuff.find("Regle")
```

```
    a3 = strbuff.find("LSysteme")
```

```
    a4 = strbuff.find("Generer")
```

```
    a5 = strbuff.find("Fonction")
```

```
    autre = strbuff.find("nouveauCaractere")
```

```
    if autre != -1:
```

```
        a1 = -1
```

```
    autre = strbuff.find("nouvelleRegle")
```

```
    if autre != -1:
```

```
        a2 = -1
```

```
    strbuff = strbuff.replace("\t", "")
```

```
    if boolL == True and a2 != -1:
```

```
        a2 = -1
```

```
    if a1 != -1:
```

```

strbuff = strbuff.replace("Caractere(", "")
chara = ""
chara = strbuff[0]
strbuff = strbuff.replace(chara, "", 1)
strbuff = strbuff.replace(" ", "")
strbuff = strbuff.replace("=", "")
strbuff = strbuff.replace("))", "")
action = ""
boolPar = False
for element in strbuff:
    if element == '(':
        boolPar = True
    if boolPar == False:
        action += element
strbuff = strbuff.replace(action, "")
fctvar = ""
fctvar = strbuff.replace("(", "")
option = ""
fonction = ""
if fctvar.isdigit():
    option = str(fctvar)
    fonction = "0"
else :
    option = "0"
    fonction = str(fctvar)

compiler.write(chara)
compiler.write(' = Caractere("")')
compiler.write(chara)
compiler.write(", ")
compiler.write(action)
compiler.write(", ")
compiler.write(option)
compiler.write(", ")
compiler.write(fonction)
compiler.write(""))
compiler.write("\n")

```

elif a2 != -1:

```

strbuff = strbuff.replace("Regle(", "")
nomVariable = ""
boolPar = False
for element in strbuff:
    if element == " " or element == "=":
        boolPar = True
    if boolPar == False:
        nomVariable += element
strbuff = strbuff.replace(nomVariable, "")
strbuff = strbuff.replace(" ", "")
strbuff = strbuff.replace("=", "")
chara = ""
chara = strbuff[0]

```

```

strbuff = strbuff.replace(chara, "", 1)
strbuff = strbuff.replace(">", "")
strbuff = strbuff.replace(")", "")

compiler.write(nomVariable)
compiler.write(" = Regle(")
compiler.write(chara)
compiler.write(", ")
tableauChara = []
for element in strbuff:
    if element == "+":
        tableauChara.append("PLUS")
    elif element == "-":
        tableauChara.append("MOINS")
    elif element == "|":
        tableauChara.append("BARRE")
    elif element == "[":
        tableauChara.append("OUVERT")
    elif element == "]":
        tableauChara.append("FERME")
    else:
        tableauChara.append(element)
for i in range(0, len(tableauChara)-1):
    compiler.write(tableauChara[i])
    compiler.write(", ")
compiler.write(tableauChara[len(tableauChara)-1])
compiler.write(")]\n")

elif a3 != -1 or boolL == True:
    boolL = True
    L1 = strbuff.find("LSysteme")
    if L1 != -1:
        strbuff = strbuff.replace("LSysteme(", "")
        strbuff = strbuff.replace(")", "")
        nomVariable = str(strbuff)
    L2 = strbuff.find("Alphabet")
    if L2 != -1:
        strbuff = strbuff.replace(" ", "")
        strbuff = strbuff.replace("Alphabet=", "")
        alphabet = str(strbuff)
    L3 = strbuff.find("Axiome")
    if L3 != -1:
        strbuff = strbuff.replace(" ", "")
        strbuff = strbuff.replace("Axiome=", "")
        tableauChara = []
        for element in strbuff:
            if element == "+":
                tableauChara.append("PLUS")
            elif element == "-":
                tableauChara.append("MOINS")
            elif element == "|":
                tableauChara.append("BARRE")

```



```

        elif element == "[":
            tableauChara.append("OUVERT")
        elif element == "]":
            tableauChara.append("FERME")
        else:
            tableauChara.append(element)
L4 = strbuff.find("Regles")
if L4 != -1:
    strbuff = strbuff.replace(" ", "")
    strbuff = strbuff.replace("Regles=", "")
    regles = str(strbuff)
L5 = strbuff.find("Pas")
if L5 != -1:
    strbuff = strbuff.replace(" ", "")
    strbuff = strbuff.replace("Pas=", "")
    pas = str(strbuff)
L6 = strbuff.find("Angle")
if L6 != -1:
    strbuff = strbuff.replace(" ", "")
    strbuff = strbuff.replace("Angle=", "")
    angle = str(strbuff)
if strbuff == "}":
    boolL = False
    compiler.write(nomVariable)
    compiler.write(" = LSysteme(")
    compiler.write(alphabet)
    compiler.write(", [")
    for i in range(0, len(tableauChara)-1):
        compiler.write(tableauChara[i])
        compiler.write(", ")
    compiler.write(tableauChara[len(tableauChara)-1])
    compiler.write("], ")
    compiler.write(regles)
    compiler.write(", ")
    compiler.write(pas)
    compiler.write(", ")
    compiler.write(angle)
    compiler.write(")\n")

elif a4 != -1 or boolGenerer == True:
    boolGenerer = True
    L1 = strbuff.find("Generer")
    if L1 != -1:
        strbuff = strbuff.replace("Generer(", "")
        strbuff = strbuff.replace(")", "")
        nomVariable = str(strbuff)
    L2 = strbuff.find("N")
    if L2 != -1:
        strbuff = strbuff.replace(" ", "")
        strbuff = strbuff.replace("N=", "")
        varN = str(strbuff)
    L3 = strbuff.find("Position")

```

```

if L3 != -1:
    compteurPos += 1
    if compteurPos == 1:
        positionTab = []
        pixelsTab = []
    strbuff = strbuff.replace(" ", "")
    strbuff = strbuff.replace("Position=", "")
    boolPar = False
    position = ""
    for element in strbuff:
        if element == ",":
            boolPar = True
        if boolPar == False:
            position += element
    positionTab.append(position)
    strbuff = strbuff.replace(position, "")
    strbuff = strbuff.replace(",", "")
    pixels = str(strbuff)
    pixelsTab.append(pixels)
if strbuff == "}":
    boolGenerer = False
    compiler.write(nomVariable)
    compiler.write(".generer()")
    compiler.write(varN)
    compiler.write(")\n")
    for i in range(compteurPos):
        compiler.write(nomVariable)
        compiler.write(".position()")
        compiler.write(positionTab[i])
        compiler.write(", ")
        compiler.write(pixelsTab[i])
        compiler.write(")\n")
    compiler.write(nomVariable)
    compiler.write(".produire()\n")

```

```

elif a5 != -1:
    strbuff = strbuff.replace("Fonction(", "")
    strbuff = strbuff.replace(")", "")
    strbuff = strbuff.replace(" ", "")
    nomVariable = ""
    boolPar = False
    for element in strbuff:
        if element == "=":
            boolPar = True
        if boolPar == False:
            nomVariable += element
    strbuff = strbuff.replace(nomVariable, "", 1)
    strbuff = strbuff.replace("=", "")
    chaineNombre = ""
    boolPar = False
    for element in strbuff:
        if element == "n":

```

```

        boolPar = True
    if boolPar == False:
        chaineNombre += element
        coefficientDirecteur = int(chaineNombre)
        strbuff = strbuff.replace(chaineNombre, "", 1)
        strbuff = strbuff.replace("n", "")
        ordonneeAorigine = int(strbuff)

        compiler.write(nomVariable)
        compiler.write(" = Fonction(")
        compiler.write(str(coefficientDirecteur))
        compiler.write(", ")
        compiler.write(str(ordonneeAorigine))
        compiler.write(")\n")

```

```

contenu = contenu.replace(removeBuff, "", 1)
strbuff = ""

```

```

compiler.write("mainloop()")

```

```

compiler.close()
mon_fichier.close()
input()

```

## Caractere.py

```

from turtle import *
from time import time

```

```

NUL = 0 ; FORWARD = 1 ; BACKWARD = 2 ; LEFT = 3 ; RIGHT = 4 ; ROLL = 5 ; SAVE = 6 ; LOAD = 7
DROITE = 0; GAUCHE = 1; HAUT = 2 ; BAS = 3

```

```

global PLUS
global MOINS
global BARRE
global OUVERT
global FERME
global Constantes

```

```

class Caractere :
    """Classe contenant un caractere, ainsi que son action associee."""

    def __init__(self, caractere, action, option, fonction):
        self.caractere = caractere
        self.action = action
        self.option = option
        self.fonction = fonction

    def afficher(self):
        """Affiche les caracteristiques de l'objet."""

```

```

print("\t", self.caractere, " : ", end="")
if self.action == 0:
    print("\tRien")
elif self.action == 1:
    print("\tAvancer")
elif self.action == 2:
    print("\tReculer")
elif self.action == 3:
    print("\tTourner a gauche")
elif self.action == 4:
    print("\tTourner a droite")
elif self.action == 5:
    print ("\tPivoter à 180 degrés")
elif self.action == 6:
    print("\tSauvegarder sa position")
elif self.action == 7:
    print ("\tRestaurer sa position")

```

class Regle :

```

"""Classe contenant une regle : qui indique de transformer un caractere en un autre."""
def __init__(self, debut, fin):
    self.debut = debut
    self.fin = fin

def afficher(self):
    """Affiche les caracteristiques de l'objet"""
    print("\t", self.debut.caractere, " ---> ", end="")
    for elemento in self.fin :
        print(elemento.caractere, end="")
    print()

```

class LSysteme :

```

"""Classe qui contient le L-Systeme.

```

Alphabet : Ensemble des caracteres utilises pour le L-Systeme [Caractere\*]

Axiome : Chaine pour n = 0 [Caractere\*]

Regles : Regles appliquees au L-Systeme [Regle\*]

Pas et angle correspondent a ceux utilises lors de la figure [int]

Fonctionnement :

Commencez par initialiser le système.

Ensuite, generez le avec une valeur n correspondant au niveau du L-systeme

Enfin, affichez le avec produire

```

"""

```

```

def __init__(self, alphabet, axiome, regles, pas, angle):
    self.alphabet = []
    self.alphabet.extend(alphabet)
    self.axiome = axiome

```

```

        self.regles = []
        self.regles.extend(regles)
        self.pas = pas
        self.angle = angle
        self.carChaine = ""
        self.genN = 0

def nouveauCaractere(self, caractere):
    """Ajoute un nouveau caractere a l'alphabet"""
    self.alphabet.extend(caractere)

def nouvelAxiome(self, axiome):
    """Remplace l'axiome precedent par un nouveau"""
    self.axiome = axiome

def nouvelleRegle(self, regle):
    """Ajoute une regles a celles deja indiquees"""
    self.regles.extend(regle)

def nouveauPas(self, pas):
    """Definit un nouveau pas pour la figure"""
    self.pas = pas

def nouvelAngle(self, angle):
    """Definit un nouvel angle pour la figure"""
    self.angle = angle

def afficher(self):
    """Affiche les informations de l'objet"""
    print(self)
    print("Alphabet :")
    for element in self.alphabet :
        element.afficher()
    print("Axiome :\n\t", end="")
    for element in self.axiome :
        print(element.caractere, end="")
    print("\nRegles :")
    for element in self.regles :
        element.afficher()
    print("Pas : ", self.pas, "px")
    print("Angle : ", self.angle, "°\n")

def generer(self, n):
    """Cree la chaine de caractere du L-systeme a la n-ieme etape"""
    testRegle = False
    chaine = ""
    chaine2 = ""
    for element in self.axiome:
        chaine += str(element.caractere)
    for i in range(n):
        for caractereActuel in chaine:
            for regleAsuivre in self.regles:

```

```

        if caractereActuel == regleAsuivre.debut.caractere:
            testRegle = True
            for element in regleAsuivre.fin:
                chaine2 += str(element.caractere)
        if testRegle == False:
            chaine2 += str(caractereActuel)
        testRegle = False
        chaine = str(chaine2)
        chaine2 = ""
        self.carChaine = str(chaine)
        self.genN = n

def produire(self):
    """Affiche la figure produite par le L-systeme"""
    speed(0)
    hideturtle()
    angleActuel = 0
    pileDeCoordonnees = []
    pileAngles = []
    for carActuel in self.carChaine:
        alphabetConstantes = self.alphabet + Constantes
        for ordreActuel in alphabetConstantes:
            if carActuel == ordreActuel.caractere:
                if ordreActuel.action == 1 and ordreActuel.option != 0 and
ordreActuel.fonction == 0:
                    forward(ordreActuel.option)
                if ordreActuel.action == 1 and ordreActuel.option == 0 and
ordreActuel.fonction == 0:
                    forward(self.pas)
                if ordreActuel.action == 1 and ordreActuel.option == 0 and
ordreActuel.fonction != 0:
                    forward(ordreActuel.fonction.calculer(self.genN))

                if ordreActuel.action == 2 and ordreActuel.option != 0 and
ordreActuel.fonction == 0:
                    backward(ordreActuel.option)
                if ordreActuel.action == 2 and ordreActuel.option == 0 and
ordreActuel.fonction == 0:
                    backward(self.pas)
                if ordreActuel.action == 2 and ordreActuel.option == 0 and
ordreActuel.fonction != 0:
                    backward(ordreActuel.fonction.calculer(self.genN))

                if ordreActuel.action == 3 and ordreActuel.option == 0 and
ordreActuel.fonction == 0:
                    left(self.angle)
                    angleActuel += self.angle
                if ordreActuel.action == 3 and ordreActuel.option != 0 and
ordreActuel.fonction == 0:
                    left(ordreActuel.option)
                    angleActuel += ordreActuel.option
                if ordreActuel.action == 3 and ordreActuel.option == 0 and

```

```

ordreActuel.fonction != 0:
    left(ordreActuel.fonction.calculer(self.genN))
    angleActuel += ordreActuel.fonction.calculer(self.genN)

    if ordreActuel.action == 4 and ordreActuel.option == 0 and
ordreActuel.fonction == 0:
        right(self.angle)
        angleActuel -= self.angle
    if ordreActuel.action == 4 and ordreActuel.option != 0 and
ordreActuel.fonction == 0:
        right(ordreActuel.option)
        angleActuel -= ordreActuel.option
    if ordreActuel.action == 4 and ordreActuel.option == 0 and
ordreActuel.fonction != 0:
        right(ordreActuel.fonction.calculer(self.genN))
        angleActuel -= ordreActuel.fonction.calculer(self.genN)

    if ordreActuel.action == 5:
        right(180)
        angleActuel += 180
    if ordreActuel.action == 6:
        pileDeCoordonnees.append(position())
        pileAngles.append(angleActuel)
    if ordreActuel.action == 7:
        allerA = pileDeCoordonnees.pop()
        up()
        goto(allerA[0], allerA[1])
        down()
        nouvelAngle = pileAngles.pop()
        right(angleActuel)
        left(nouvelAngle)
        angleActuel = nouvelAngle

def position(self, position, pixels): #Il faudrait obtenir la taille de l'écran
    """Definit la position de l'origine de la figure tracee

    position : indique l'endroit ou doit etre positionne l'origine
    pixels : indique a quelle distance en pixels l'origine doit etre par rapport a la position
indiquee

    """
    speed(0)
    hideturtle()
    up()
    if position == 0:
        forward(759-pixels)
    elif position == 1:
        backward(765-pixels)
    elif position == 2:
        left(90)
        forward(399-pixels)

```

```

        right(90)
    elif position == 3:
        left(90)
        backward(391-pixels)
        right(90)
    down()

```

class Fonction:

```

    """Définit une fonction"""

```

```

    def __init__(self, coefficientDirecteur, ordonneeAOrigine):

```

```

        self.coefficientDirecteur = coefficientDirecteur

```

```

        self.ordonneeAOrigine = ordonneeAOrigine

```

```

    def calculer(self, n):

```

```

        return self.coefficientDirecteur*n+self.ordonneeAOrigine

```

```

PLUS = Caractere("+", LEFT, 0, 0)

```

```

MOINS = Caractere("-", RIGHT, 0, 0)

```

```

BARRE = Caractere("|", ROLL, 0, 0)

```

```

OUVERT = Caractere("[", SAVE, 0, 0)

```

```

FERME = Caractere("]", LOAD, 0, 0)

```

```

Constantes = [PLUS, MOINS, BARRE, OUVERT, FERME]

```

## batch.bat

Start compilateur.py

## batch2.bat

Start compile.py

## Diagramme du fonctionnement du programme



