SIMULATION DE FEUX DE FORÊT
(en langage Python)

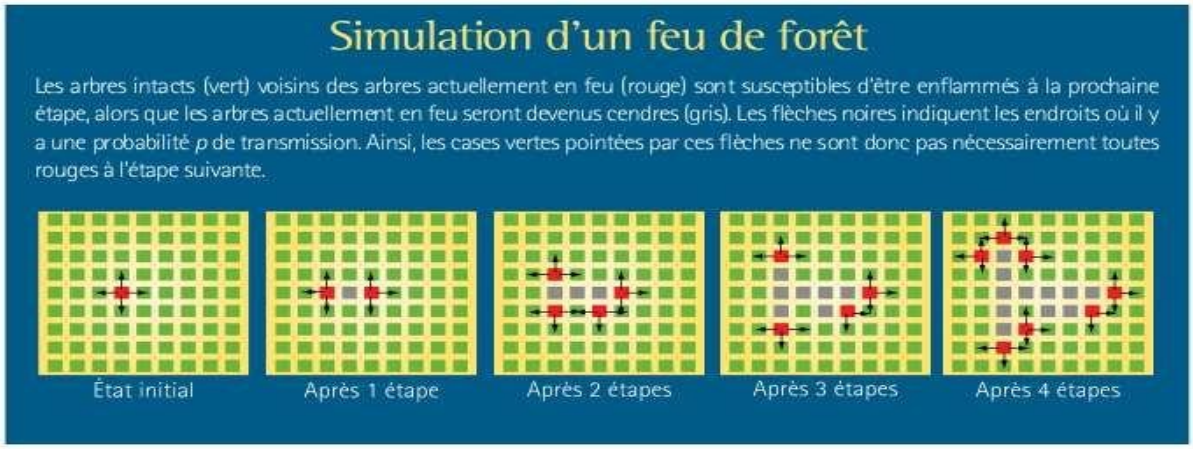
Pour le projet que nous devons présenter au bac, nous avons modélisé la propagation d'un feu de forêt. Pour cela, nous avons dû réfléchir sur:

1- la structure de la forêt
2- la structure du feu (c-à-d la propagation du feu dans toutes les directions)

À travers un grand choix de programmes que l'on a trouvé, par exemple la simulation grâce à un automate cellulaire, en utilisant un module graphique comme Pygame etc... Nous en avons sélectionné seulement deux:

- Dans le premier programme, on peut créer une centaine d'images qui nous montrent la progression d'un feu de forêt.
  Le feu se propage de manière aléatoire et l'arbre qui était en feu devient en cendre (comme sur la photo ci-dessous).

- Dans le second programme, la forêt sera basée sous la forme d'une matrice. Les arbres et le point de départ de la propagation du feu sont placés aléatoirement dans la matrice.



Le cadre du projet est :

- La salle de cours avec des ordinateurs à disposition pour les recherches.
- Un groupe de 4 élèves dont une passant en candidate libre : Oummeymane ADAM, Sandra VELLAYOUDOM, Shane MAMODALY LADA, Emilie LAFOSSE.
- Un groupe sur Google Groups (adresse du Groupe) pour pouvoir communiquer avec notre professeur et de nombreux échanges par mails (nous étions tous inscrits sur Gmail) et par Facebook et Skype pour les membres du groupe.

Les groupes ainsi que les projets ont été choisis et attribués par notre professeur.

Une fois notre groupe constitué, nous avons commencé les recherches sur la simulation d'un feu de forêt afin de comprendre son concept.

Dans un premier temps, les recherches ont été réalisées séparément, chacun de son côté, chez soi pour pouvoir mettre en commun quand nous nous retrouvions pour les cours, dans le but de faire travailler chaque membre du groupe sur des parties spécifiques tout en prenant connaissance des autres parties du projet.

Nous avons eu beaucoup de difficultés surtout en ce qui concerne l'écriture du programme pour obtenir la simulation.

Notre professeur nous a autorisé à utiliser et modifier des programmes obtenus sur Internet à condition de citer la source.

Nous avons trouvé beaucoup d'exemples en faisant des recherches sur internet, puis nous avons sélectionné ceux qui selon nous étaient les plus simples et les plus intéressants afin d'obtenir un résultat convenable et compréhensible avec notre niveau de programmation.

Nous envoyions régulièrement des bilans de notre travail et de notre avancement à notre professeur afin qu'elle puisse être témoin de l'évolution de nos recherches et qu'elle soit en mesure de nous donner des conseils spécifiques à notre groupe et de nous orienter vers des sites à étudier plus en profondeur pour avoir un projet solide pour notre présentation.

Notre professeur communiquait toujours avec nous via Gmail et Google Groups où elle avait mis en place en début d'année un groupe comportant les élèves de la spécialité pour permettre une communication permanente et instantanée avec toute la classe d'ISN.

En dehors des heures de cours effectuées, nous communiquions grâce aux réseaux sociaux comme Facebook et Skype pour pouvoir faire des visio-conférences et rester en contact.

Au début nous avions l'intention d'utiliser un automate cellulaire grâce aux modules graphiques comme Numpy ou encore Pygame mais nous avons abandonné à cause du niveau requis en programmation trop élevé par rapport au notre.

Un automate cellulaire est une grille régulière de «cellules» contenant chacune un «état» choisi parmi un ensemble fini et qui peut évoluer au cours du temps.
L'état d'une cellule au temps *t+1* est fonction de l'état au temps *t* d'un nombre fini de cellules appelé son «voisinage». À chaque nouvelle unité de temps, les mêmes règles sont appliquées simultanément à toutes les cellules de la grille, produisant une nouvelle «génération» de cellules dépendant entièrement de la génération précédente.

Nous avons travaillé en Python dans une interface IDLE.

IDLE signifie "Integrated DeveLopment Environment" (Environnement de développement intégré).
C'est un logiciel qui nous permet d'écrire des programmes en langage Python.
Les principales fonctionnalités de IDLE sont:

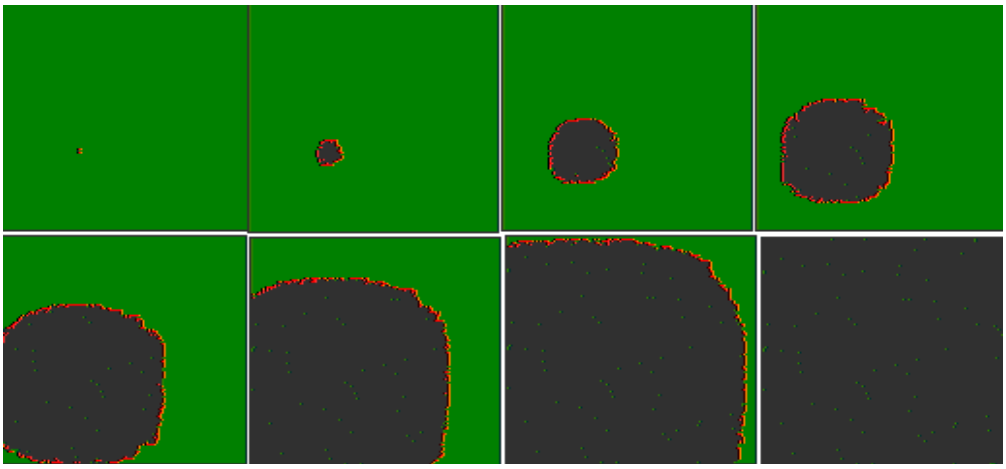
- l'éditeur texte avec coloration syntaxique (qui consiste à formater automatiquement chacun des éléments du texte affiché en utilisant une couleur caractéristique de son type.)
- le terminal Python avec coloration syntaxique
- le débogueur intégré avec avancement par étape, point d'arrêts persistants et pile d'appels.

1. La première simulation est un programme nous permettant de créer une centaine d'images dans un même répertoire qui nous montre la progression d'un feu de forêt.
Il fonctionne sur IDLE python 2.7

* Nous avons changé certains paramètres dans le code source car ils n'étaient pas adéquats à la situation: - la forêt qui était bleu nous l'avons mise en vert
- le feu en vert nous l'avons mis en rouge
- les cendres en gris etc....

CODAGE
arbre =====> 1pixel vert
arbre en feu =====> 1 pixel rouge
cendre =====> 1 pixel gris
Pas d'obstacle

Exemple d'une sortie de ce programme



* Cependant la propagation du feu est trop parfaite avec les paramètres que nous avons utilisés : le feu s'étend de manière équilibrée partout tandis que si nous comparons cette simulation à d'autres simulations vidéos nous pouvons voir que les autres simulations sont beaucoup plus réalistes.

* De plus, grâce à ces centaines d'images nous avons réussi à monter une petite vidéo. Au début nous avions fait un petit film avec les images que nous avions obtenu à partir du programme original.
Mais comme les images étaient trop petites, nous avons modifié leurs résolutions pour que la vidéo ne soit pas pixelisée en plein écran.
Nous avons créé la vidéo en utilisant une commande dans le Terminal (sous Ubuntu):

```
ffmpeg -f image2 -i image%d.jpg video.mpg
```

Pour le premier programme, le code source se trouvait dans un fichier PDF sur Internet. La difficulté première fut de corriger la syntaxe du code source de ce programme car en effectuant un simple copier-coller, nous perdions toute la structure syntaxique du programme (notamment avec l'apparition d'espaces entre les caractères rendant le code invalide). Il a donc fallu trouver les retours chariots, refaire toutes les identations (elles sont obligatoires pour le langage Python afin de former des blocs ou des lignes valides), ce qui fut déjà très long, avant que le programme ne marche dans l'IDLE.
Cela nous a obligé à identifier où étaient les fonctions et essayer de comprendre ce qu'elles faisaient.

2. Le second programme fonctionne sur IDLE python 3.2

* La forêt, les obstacles, le feu sont représentés par des lettres.
La propagation du feu se fait horizontalement et verticalement.

* La simulation à partir du source original est correcte mais le problème réside au niveau de l'esthétique.
Nous avons donc appliqué certaines modifications comme:

- mettre des espaces entre les lettres
- mettre la simulation dans une sorte de cadre
- nous pouvons aussi changer la structure, c-à-d agrandir la forêt, limiter la propagation du feu en isolant la forêt par des obstacles etc...

CODAGE

arbre =====> A
arbre en feu =====> #
obstacle =====> T
Pas de cendres

Algorithme

Dans un tableau/matrice 10*5
Placement aléatoire de A et de T
1<x<10 et 1<y<5
Apparition de manière aléatoire d'un # dans [x,y]
  Tant qu'il y a une case A voisine de # faire:
    A t+1 vérifier les cases du tableau
      Si la case A est voisine de # alors la case = #
      Sinon
        Si la case T est voisine de # alors case reste T
        Fin Si
      Fin Sinon
      Fin Si
    t+1 → t
  Fin Tant que

Exemple d'une sortie de ce programme

Nous nous sommes donc concentrés sur ce second programme pour plusieurs raisons:

- le code source est moins complexe à analyser et moins long.
- nous pouvons le modifier aisément de façon à rentrer nos propres paramètres, ce qui est beaucoup plus simple que dans le premier programme, vu notre niveau actuel en programmation.

Pour conclure, il faut savoir que les programmes de simulation que nous avons présentés sont plutôt limités esthétiquement et fonctionnellement. Afin d'améliorer notre programme, nous aurions voulu le modéliser à l'aide d'une interface graphique sous forme de plate-forme de simulation 2D grâce à Pygame et d'y ajouter des fonctionnalités telles que la possibilité de choisir le point de départ du feu, mais par manque de temps et surtout par manque d'expérience nous en sommes restés à un niveau très modeste.

Bibliographie

→ http://www.irem.univ-paris-diderot.fr/up/memoiremodelisation-1.pdf
→ http://fr.wikipedia.org/wiki/Automate_cellulaire
→ http://www.irem.univ-paris-diderot.fr/articles/feux_de_foret
→ https://deptinfo-ensip.univ-poitiers.fr/ENS/doku/doku.php?id=tp:automates_cellulaires_2d

→ http://www.siteduzero.com/informatique/tutoriels/interface-graphique-pygame-pour-python

› Exemples de commandes du module Numpy:

```
import numpy as np       # importation du module dans le programme Python
t=np.zeros((5,5))        # Tableau nul de 5 cases par 5
t[2,3]=42                # exemple de modification d'une case
```

› Exemples de commandes du module Pygame:

```
import pygame
pygame.init()
xx,yy=200,200      # initialisation
```