Lenglemetz Zoé
Duchez Rémi
Casandjian Ewan

Dossier projet : ISN

Jeu de la vie

Année 2017-2018

Au début, l'enjeu principal de notre projet était de réussir à allier l'art et l'informatique mais des aspects techniques nous ont contraints à revoir cette idée. En effet, notre idée de base était de crée un cube de LED au format 8*8*8. Mais, le coté électronique, et plus particulièrement l'alimentation du cube et de ses différents étages, à représenter un problème majeur. Ainsi voyant que nous n'arrivions pas à résoudre ce problème, nous avons décidé de changer de projet.

Nous avons donc choisis de plutôt allier un aspect mathématique, ici la logique, avec l'informatique. Dans un désir de rester dans une approche ludique, nous avons entamé des recherches pour trouver des jeux correspondant à nos critères.

Dans un premier temps, nous avons d'abord pensé à des jeux tels que les échecs, les dames et le jeu de la vie.

Mais en prenant un peu de recul, nous avons compris que l'intelligence artificielle nécessaire pour créer un jeu de dame ou d'échecs était très complexe.

Notre choix s'est donc porter sur le jeu de la vie. En effet, ce jeu répondait à notre désir d'allier les mathématiques à l'informatique, tout en ayant la possibilité de coder en python.

Le jeu de la vie est un automate cellulaire pensé vers 1970 par un certain John Horton Conway. Depuis ce jour-là et encore aujourd'hui, le jeu de la vie reste l'un des automates cellulaire les plus connus.

Mais qu'est-ce qu'un automate cellulaire ?

Un automate cellulaire peut être considéré comme un objet mathématique et informatique qui évolue étapes par étapes selon des règles définie. Cette évolution est, dans une certaine mesure, une imitation des capacités autoreproductrices que possèdent les cellules constituant les êtres vivants.

Les règles définissant un automate cellulaire peuvent être diverses et variés et selon ces dernières différents dessins apparaissent de par la coloration des cellules en fonction de leur état.

Ainsi, on peut donc dire que le jeu de la vie, n'est en réalité, pas considéré comme un véritable jeu car il ne nécessite pas la présence d'un joueur pour fonctionner. Le principe est simple, l'automate est constitué d'une grille dont les cases sont appelées « cellules », ces dernières peuvent prendre deux états différents, «mort» ou «vivant».

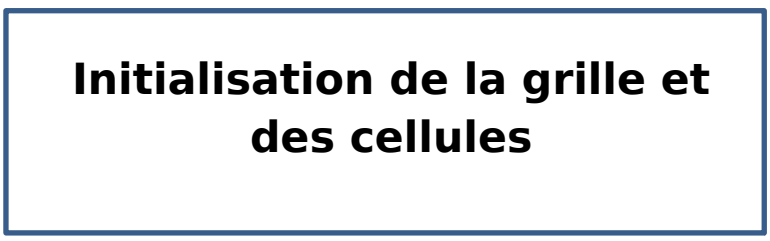
La règle que nous avons décidé d'appliquer était que chaque cellule vivante ne possédant pas au moins deux voisins devait mourir tandis qu'une cellule morte présentant au moins deux voisins vivant devenait vivante à son tour. Mais le changement d'état des cellules étant automatique, l'interaction avec la personne était très faible voir quasi-inexistante.

Cependant, nous voulions faire en sorte qu'il y ait un minimum d'interaction pour rendre la chose plus attrayante. Pour cela nous avons décidé de laisser le choix des cellules de départ à l'utilisateur et non d'utiliser des cellules prédéfinies. De plus, ce dernier peut à tout moment réinitialiser la grille et les cellules.

On peut ainsi considérer que le principe des automates cellulaires constitue le squelette de notre projet. Cependant il fallait maintenant le travailler et le mettre en place pour en arriver petit à petit au jeu de la vie.

Pour coder ce projet, nous avons donc utilisé le langage python. Ce dernier étant relativement simple, possédant une très grande bibliothèque et pouvant être utilisé par de nombreuses applications, il convenait parfaitement à notre projet.

Schéma du jeu de la vie :



Lancement de l’automate cellulaire

L’état des cellules dépend des règles prédéfini. Ici, une cellule morte possédant exactement trois voisines vivantes devient vivante tandis qu’une cellule vivante possédant au moins deux voisines vivantes

La boucle sur laquelle est basé l’algorithme se répète tant que cela est possible

Répartition des taches :

Fonction objet	Tkinter	Algorithme	Incrémentation	Rédaction du dossier

Zoé	Zoé	Zoé		
	Rémi	Rémi		Rémi
			Ewan	Ewan

Démarche collaborative :

Pour créer notre projet nous avons utilisé différentes plateformes collaboratives dont Github et Google doc.

La plateforme Github nous a permis de remplir un cahier de bord et de nous partager nos recherches, les besoins que nous avions pour compléter notre code et enfin de consulter l'avancement du projet et de suivre l'évolution du code.

La plateforme Google doc quant à elle nous a servi pour la rédaction du dossier et le coté algorithmique du projet

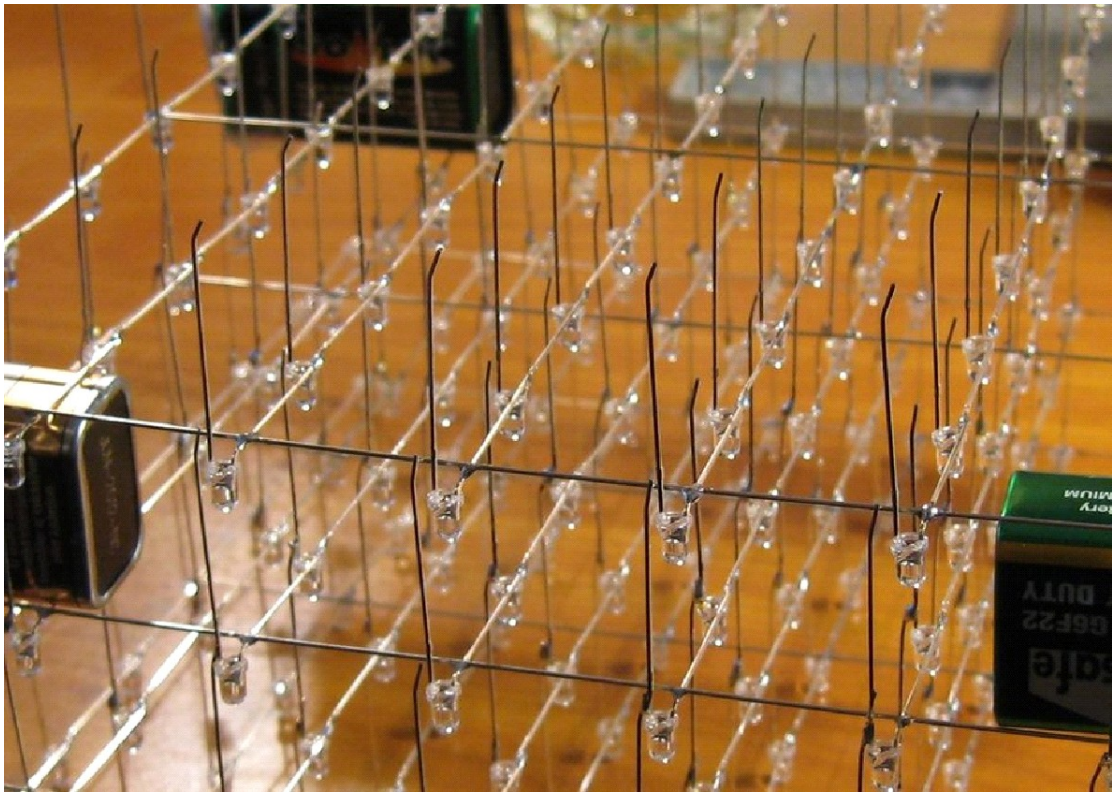
Ainsi nous pouvions suivre l'avancement du projet et rajouter petit à petit des choses au code ou corriger des erreurs qui empêchaient le fonctionnement de ce dernier. Nous avons de plus, profité des cours d'ISN pour nous retrouver et en discuter.

Compte rendu personnel : Zoé

A la base notre projet était d'allier le code au côté esthétique grâce à un cube de led 8*8*8

Nous avons donc commandé près de 1000 led sur internet et j'ai commencé à souder les leds ensemble

Cela s'est avéré bien plus dur que prévus une fois qu'il a fallu attacher les étages ensemble car l'accès au centre du cube était très compliqué.

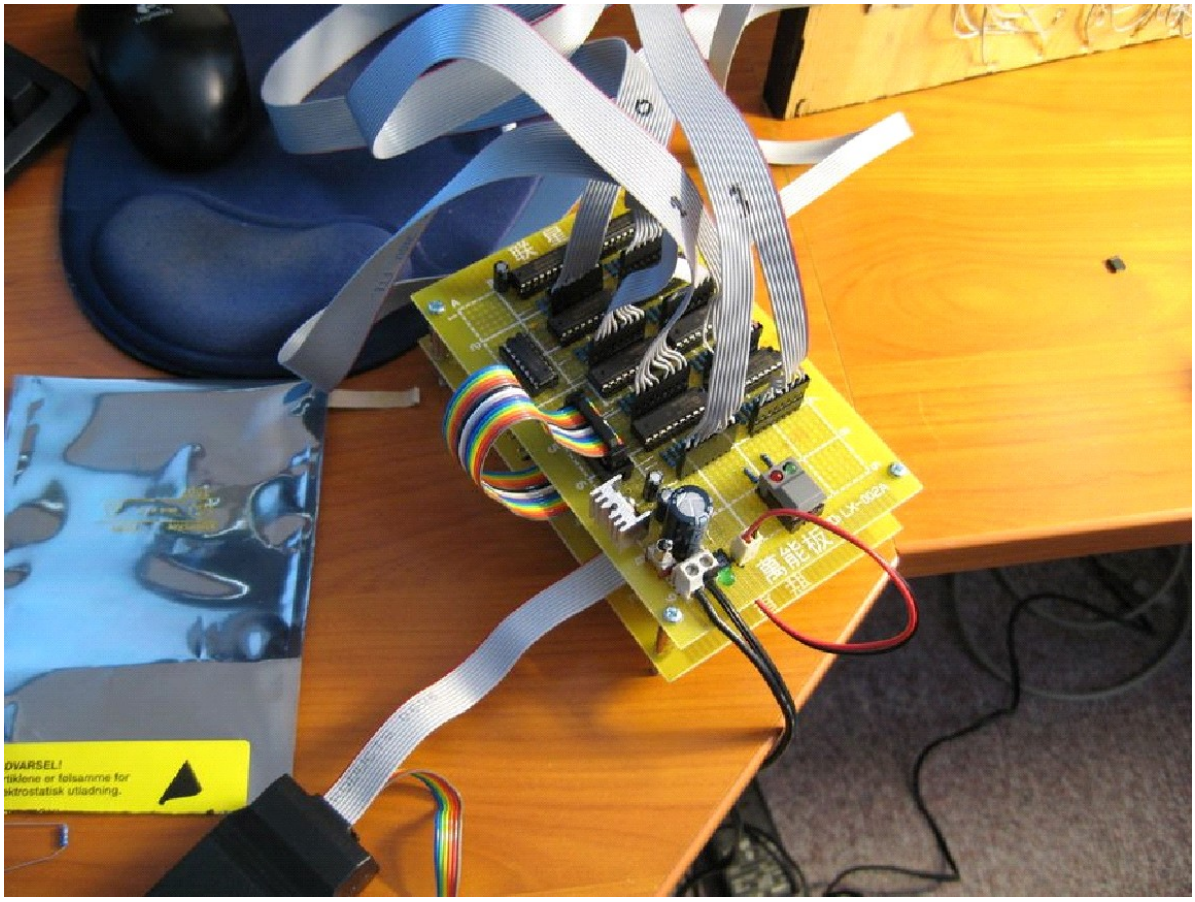


Puis nous nous sommes procuré un raspberry pi afin de pouvoir allier le software au hardware. Et c'est là que nous avons rencontré d'énormes difficultés

En effet nous ne connaissons presque rien en électronique et nous avions oublié de penser aux compétences requise pour un tel projet en effet il faut réussir à gérer les résistances et les sources d'énergie car les leds ont deux côtés l'anode (+) et la cathode (-). Or le câblage de tout cela est très complexe et la moindre erreur peut faire griller toutes les leds. De plus je n'avais aucune idée de comment gérer le problème du câblage car la raspberry pi ne possédait que 12 ports et nous avions plus de 64 fils.

J'ai essayé de trouver une solution sur internet mais impossible d'en trouver une simple et qui requiert peu de matériel et de compétences.

Nous avons donc abandonné le côté hardware pour nous concentrer sur le côté software



Idéalement et si nous avions réussi le dessous du cube aurait été comme ceci.

Nous avons ensuite entamé de coder un automate cellulaire de Conway.

J'ai eu pour rôle de faire le squelette du programme. La principale difficulté de ce programme est le fait qu'il doit stocker les données et tout exécuter en même temps sinon les cases changent en différé au lieu de changer simultanément.

Au départ je voulais utiliser un système où il y a deux univers:

-le premier est l'univers où il y a les états de chaque cellule avant l'analyse

-et un second ou le programme aurais changé en fonction du premier univers et sans prendre en compte les précédents calculs l'état des cellules

Cependant en cherchant un peu j'ai trouvé un moyens beaucoup plus simple qui est l'utilisation de la fonction orientée objet : class (un objet est une variable pouvant contenir elles-mêmes des fonctions et variables.)

Elle permet ici de définir le jeu de la vie avec plusieurs caractéristiques comme un tableau de cases soit noir soit blanche etc...

Au début j'ai eu du mal a la prendre en mains car il y a beaucoup de syntaxe a respecter afin qu'elle soit fonctionnelle comme self etc...

Sur cette photo on peut voir l'ensemble des objets définis.

J'ai également fais la partie tkinter du programme donc l'interface graphique qui s'intègre dans mes objets.

J'ai tout d'abord créer une fenêtre principale grâce a :Fenetre.mainloop()

On utilise un"Label", c'est-à-dire un objet graphique affichant du texte

Je crée ensuite un tableau grâce à grid() et je définie les deux états de mes cases du tableau grâce à bg (background en anglais)

Les cases à cocher du tableau sont définies dans la class Checkbutton. On utilise une variable pour surveiller la sélection de la case.et ainsi on peut savoir si la case est cochée ou non. Ici j'ai associé la couleur noir à l'état disable et blanc a normal.

```
# bouton pour lancer le jeu

self.start_button = Button(self.parent, text = "Start Game", command = self.simulate_game)

self.start_button.grid(row = 1, column = 1, sticky = E)


self.reset_button = Button(self.parent, text = "Reset", state = DISABLED, command = self.reset_game)

self.reset_button.grid(row =1 , column = 2, sticky = W)
```

Ici grâce à tkinter j'ai créé le bouton pour commencer le jeu et celui de reset.

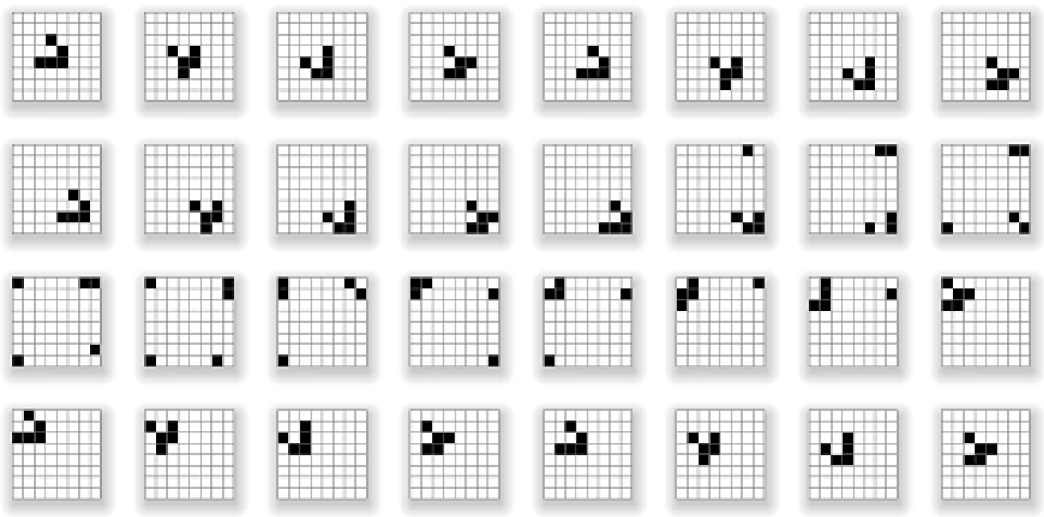
Afin de tester les différentes étapes de mon programme lors d'un bug. Je crée une nouvelle fenêtre tkinter avec écrit yo dedans et je la déplacer sur les lignes du programme lorsque cette fenêtre ne s'affichait plus cela voulait dire que le bug se situait ici

On pourrait améliorer le programme en améliorant par exemple l'interface graphique encore très archaïque. de plus on pourrait réussir à rendre possible la sélection de case durant la réalisation du programme. en effet lorsque le programme est en marche il n'est plus possible de changer manuellement l'état d'une cellule.

Ou encore en ajoutant des dessins pré-enregistrée comme le glider ou il suffira de cliquer sur un bouton pour lancer un schéma pré-enregistrée avec un développement intéressant comme ci-dessous avec un schéma appelé glitter.

Sur cette photo on peut voir l'évolution du glitter.

C'est un schéma qui oscille dans le temps tout en se déplaçant sur la grille.



Lien Github

https://github.com/lasource2018/plus_qu-un_simple_cube.../tree/code