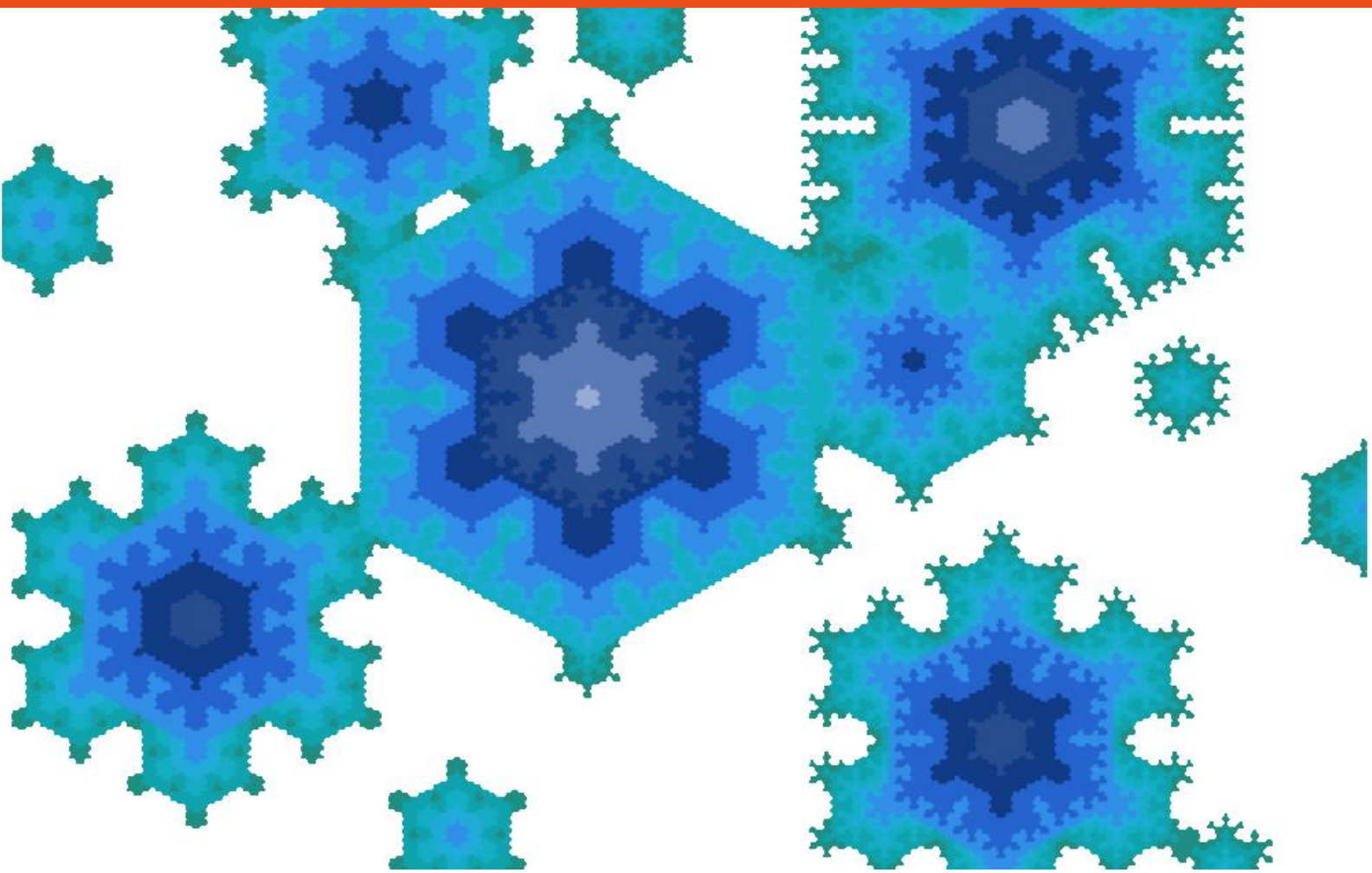
LES TROPHÉES NSI

Édition 2022

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET



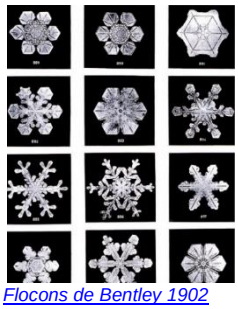
Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : SnowFake : Simuler des flocons

> PRÉSENTATION GÉNÉRALE :



Flocons de Bentley 1902

Les flocons de neige sont un des plus beaux phénomènes dans la nature. Certains font penser à un ouvrage de dentelle d'une finesse extraordinaire, chaque flocon semble unique et on observe des symétries, les branches sont au nombre de six. La beauté des flocons est encore plus incroyable, car c'est une beauté éphémère. Ils sont sources d'émerveillement des scientifiques, enfants et artistes.

Au cours de l'histoire, plusieurs d'entre eux ont été inspirés et ont cherché à capturer la beauté des flocons ; le plus célèbre d'entre eux est Wilson Bentley. Au cours de sa vie, ce dernier a pris plus de 5000 photos de flocons de neige.

Tout cela nous amène à une question centrale. **Pouvons-nous tenter de comprendre comment se forment les flocons de neige et simuler leur croissance ?**

Dans son livre La petite histoire des flocons de neige Étienne Ghys présente l'article publié par Kepler en 1611 « Un cadeau de nouvelle année à propos de la neige à six branches » où le mathématicien se questionne sur la symétrie et les six branches des flocons.

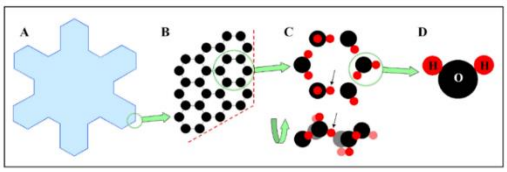
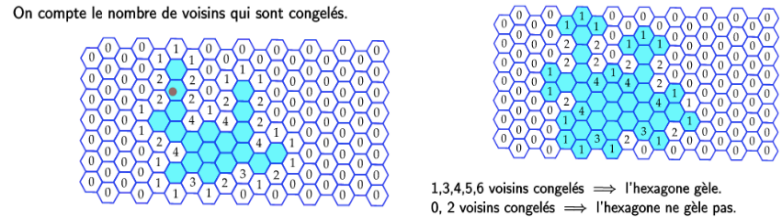


Schéma montrant la structure hexagonale d'un flocon

Aujourd'hui, grâce aux modèles physiques, nous pouvons expliquer que ceci provient des propriétés chimiques des molécules d'eau, car les flocons de neige sont des cristaux de glace. À l'état solide, les molécules d'eau s'organisent en une structure cristalline hexagonale expliquant les 6 pointes.

Un modèle mathématique de la formation des flocons, le modèle de Norman Packard est proposé dans l'ouvrage de Ghys à la page 95. On représente en bleu les hexagones congelés à l'étape n. Pour savoir si un hexagone non congelé sera congelé à l'étape $n+1$ on utilise les deux règles très simples :



Une visualisation du modèle utilisé. Source de la visualisation

Nous pouvons constater que la grille du modèle est hexagonale en raison des 6 branches des flocons. L'état d'une cellule à la prochaine étape dépend de celle de ses partenaires à l'étape précédente. Ce modèle est donc similaire au *Jeu de la Vie* de John Horton Conway créé en 1970. C'est un automate cellulaire, "un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles préétablies" (Jeu de la vie — Wikipédia). Le jeu a originellement été inventé comme un problème mathématique, mais peut aussi être utilisé comme analogie pour la vie de vraies cellules ; il est également adapté ici puisqu'la structure cristalline des flocons de neige est construite par des liens entre molécules. Nous créons des faux flocons de neige, ce qui explique le nom « SnowFake » signifiant neige fausse en anglais, inspiré du nom des DeepFakes.

L'objectif de ce projet est donc d'implémenter le modèle mathématique de Norman Packard pour simuler nos propres flocons automatiquement.

> ORGANISATION DU TRAVAIL :

Tomas McNamer : Interfaces humain-ordinateur lors de la simulation (interface PyGame) et lors de la configuration (interface CLI)

Alexander Aumais : Intégration, calculs de simulation, rédaction de l'article et autres (GIFs, découpe laser)
Katherine Langille : Étude et visualisations statistiques et production de la vidéo
Simon Ngassam : Gestion de grilles hexagonales et documentation (README, docstrings et prototypage)

Lors de la mise en place du projet, nous avons décidé que le code serait développé de façon modulaire avec des spécifications précises pour chaque partie du système. Une fois la structure complète du projet définie, ceci nous a permis d'être assez autonomes ; les tâches de programmation étaient principalement dirigées par l'individu auquel elles avaient été réparties, mais nous nous sommes tous entraidés. Le développement du projet s'est fait en partie pendant le cours de NSI – nous basculions sur le projet quand nous avions fini le travail obligatoire en attendant que les autres élèves terminent - et également en dehors des cours. Excepté les échanges pendant le cours de NSI nous n'avons pas eu de réunions formelles, mais avons utilisé Discord, une plateforme de messagerie instantanée, pour communiquer entre nous, établir les priorités et demander de l'aide.

Tout le code est stocké et partagé avec Git sur GitHub qui rend très simple la synchronisation des fichiers et structure bien l'addition de nouvelles fonctionnalités sans briser le reste du code.

La modularité du programme rend simple l'élimination de bugs dans chaque partie du programme au fur et à mesure.

> LES ÉTAPES DU PROJET :

Étape	Description	Difficultés	Durée
Idée	Le livre « La petite histoire des flocons de neige » d'Étienne Ghys est arrivé dans notre bibliothèque et nous l'avons lu dans le cadre de la fête de la science. Nous avons par la suite décidé de programmer sa simulation dans le cadre du cours de NSI.	Aucune	Quelques jours
Première version	Après avoir défini des fonctions et des classes à écrire, nous avons divisé les tâches entre nous et nous avons programmé la première version de notre simulation.	Gérer une grille hexagonale	Une semaine
Améliorations	Une fois la première version terminée, nous avons décidé d'améliorer notre programme en ajoutant l'option de créer des GIFs, l'option de passer des arguments par le terminal, études et visualisations statistiques.	Des GIFs de centaines de mégaoctets, ajouter de la compression	Quelques semaines
Finalisation et documentation	Après avoir fini toutes les grandes améliorations, nous avons écrit une documentation complète pour notre projet et nous avons rajouté des touches de finition.	Écrire la documentation facile à comprendre	Quelques semaines

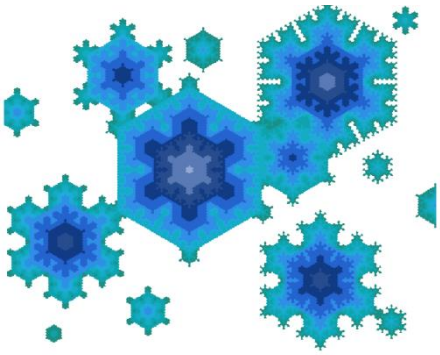
> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

La grille hexagonale correspond à la plus grande difficulté que nous avons dû gérer
Pour représenter la grille d'hexagones, nous avons utilisé un tableau à 2 dimensions. En effet, dans une grille d'hexagones, il n'est pas possible d'aligner les hexagones à la fois sur les rangées et les colonnes ; on peut seulement aligner sur l'une des deux. Vu que nous dessinons les hexagones avec deux arêtes horizontales, nous avons donc des colonnes droites. Nous représentons notre tableau à 2 dimensions comme une liste de colonnes.

Un autre problème que nous avons rencontré est le fait qu'un hexagone a 6 voisins, et non 8 (voir figure 1 dans l'annexe en fin de document). Ceci nécessite alors une modification des règles du Jeu de la Vie (qui fonctionne sur 8 cellules voisines), mais rend aussi les calculs des coordonnés des cellules voisines plus complexes et dépend de la parité des colonnes. En lire plus.

Néanmoins, **nous avons été capables de bien finir le projet** et y apporter plusieurs fonctionnalités intéressantes.

La simulation : *Lien vers une version animée*



Une image de la simulation

Le modèle de Norman créé des flocons ravissants, et nous colorions les cellules hexagonales en fonction de leur « âge » (le temps qu'elles ont été gelés). Une cellule qui existait à la première étape sera plus sombre qu'une née à la 50e (en vieillissant les couleurs deviennent plus foncée et puis vont vers le blanc). (*Voir figure 2 de l'annexe*)

Pour une vérification du fonctionnement du projet dans son intégralité, nous nous sommes tournés vers le livre qui a inspiré le projet. À la page 101 de *La petite histoire des flocons de neige*, il y a des visualisations des 50 premières étapes de son modèle. Nous avons vérifié que notre code reflète parfaitement ce comportement.

Le stockage modulaire des données des cellules.

Nous avons utilisé la programmation orientée objet pour nos cellules nous permettant d'y intégrer des métadonnées. Pour d'autres d'applications de simulations hexagonales, il est très facile d'ajouter d'autres métadonnées qui peuvent être utilisées pour certaines règles (par exemple la simulation d'autres minéraux).

L'exportation de GIFs pour le partage simple des simulations.

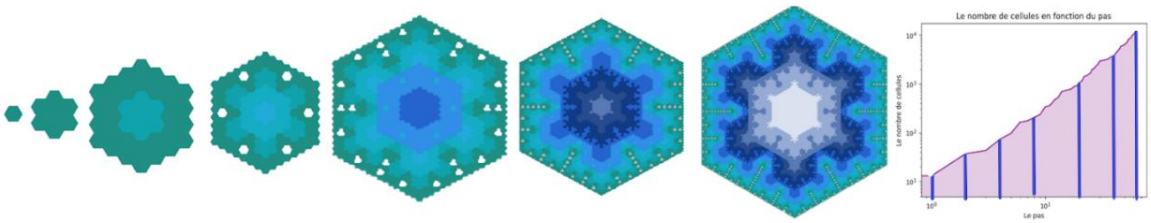
Pour partager les résultats des simulations facilement avec nos amis, nous pouvons exporter des fichiers .GIF (images animées). *Voir un exemple*

Analyses statistiques.

Nous pouvons stocker des données comme l'âge moyen des cellules vivantes, le nombre de cellules total et leur croissance sont ensuite les analyser avec des graphes grâce à la bibliothèque matplotlib.

En regardant les premières simulations, nous nous sommes demandé si nos flocons sont des fractales.

Nous avons observé d'une part que les flocons reviennent toujours à une forme presque hexagonale et d'autre part que le nombre de cellules gelées sur une échelle log-log crée une ligne presque droite.



Ces deux observations semblent indiquer qu'en utilisant le modèle de Norman nous créons des fractales en reprenant les définitions proposées par Wolfram Alpha.

Découpe laser.

Notre lycée à une découpe laser et nous avons décidé d'essayer d'exporter des fichiers pour découper et graver des morceaux de bois. Nous sommes donc capables de créer des vrais

objets à partir de ces simulations ! (*Voir figure 3 de l'annexe).* À la fin de chaque année scolaire, les élèves de terminales à notre Lycée organisent un bal pour lequel ils doivent lever des fonds, l'an prochain nous comptons produire des flocons découpés que nous vendrons décorés à noël.

> OUVERTURE :

Nous sommes très contents que le projet fonctionne bien pour nos applications et qu'il est capable de créer des belles simulations, animations et études intéressantes. Mais comme dans tout projet, il y a quelque chose que nous pourrons améliorer dans le futur.

Tout d'abord, la mise en place des **modèles** pseudo-aléatoires ou plus réalistes pour créer des flocons uniques serait une piste intéressante. Ceci peut être fait par le modèle de Clifford Reiter à la page 102 de *La petite histoire des flocons de neige* qui imite mieux la réalité.

Il aussi serait utile d'avoir des **grilles de tailles dynamiques** et un **meilleur stockage des données** pour utiliser moins de mémoire vive. Il serait aussi intéressant d'améliorer le canevas sur lequel les cellules sont affichées dans le but de pouvoir zoomer et bouger autour de la grille. Ceci facilitera l'exploration de plus grandes grilles.

Ensuite, **l'efficacité de nos algorithmes** peut encore être mise au point, possiblement en regroupant des groupes de cellules pour éviter une itération qui passe par chacune des centaines de milliers de cellules.

De plus, nous voulons améliorer **nos interfaces** afin de les rendre encore plus intuitives et esthétiques. La plateforme que nous utilisons pour l'interface graphique est simple à programmer, mais est vieille et manque des fonctionnalités d'optimisation comme l'exploitation de la carte graphique. Ce serait une grande chose que nous changerons si nous refaisons ce projet.

Enfin, en plus du travail de programmation, il serait aussi très intéressant de faire plus de **recherches mathématiques sur des jeux de la vie à partir d'une grille d'hexagones**, car la façon dont ces polygones peuvent paver le plan sans se sous diviser rend plus complexe la création de structures intéressantes comme des vaisseaux ou oscillateurs présents dans le Jeu de la Vie de Conway. Ces recherches pourraient se faire en choisissant des règles différentes de celles que nous utilisons pour les flocons.

En conclusion, le projet SnowFake a été un projet très amusant à développer après avoir fini tout le travail en NSI. Nous avons beacoup appris sur la physique, les modèles mathématiques, à propos de la programmation et puis de la communication scientifique.

Un message personnel de chaque membre :

Alex : J'ai vraiment aimé travailler dans ce projet et avec cette équipe. C'était amusant de tout planifier et organiser et la modularité de programme m'a laissé explorer librement des fonctionnalités comme la découpe laser et les animations.

Tomas : J'ai trouvé intéressant les aspects techniques de la simulation et les calculs graphiques pour l'interface utilisatrice.

Simon : Travailler sur ce projet a été une bonne opportunité pour approfondir mes connaissances en Python et j'ai bien aimé travailler en équipe.

Katherine : J'ai beaucoup appris sur les études statistiques et les différents modèles mathématiques possibles, ce qui est vraiment intéressant.

DOCUMENTATION

Installation

1. La première étape consiste à cloner le répertoire de notre code et entrer dans le dossier.

```
$ git clone https://github.com/SuperZooper3/Game-of-Hex.git
$ cd Game-of-Hex
```

2. Il faut ensuite installer les bibliothèques dont nous avons besoin.

```
$ pip install -r requirements.txt
```

Utilisation

Afin d'exécuté notre programme, il faut naviguer dans le dossier `src` et exécuté le fichier principal :

```
$ cd src/
$ python3 main.py
```

Notre programme utilise une interface utilisateur standard au-dessus de la ligne de commande (CLI). Il est possible de consulter les options d'utilisation (en anglais) directement dans la ligne de commande à l'aide de la commande `python3 main.py --help`.

La configuration du programme se fait de façon standard sur la ligne de commande. Les différentes configurations possibles sont décrites ci-bas.

Commande	Description
`python3 main.py --help`	Écrit les différentes options d'utilisation sur la console
`python3 main.py -x X -y Y`	Permet de choisir la taille de la grille, avec X le nombre de cellule sur l'axe horizontal et Y to nombre de cellule sur l'axe vertical
`python3 main.py -f F` `python3 main.py --maxfps F`	Permet de choisir le nombre maximum d'état par seconde (par exemple for ralentir la simulation).
`python3 main.py -r R` `python3 main.py --radius R`	Permet de choisir la taille d'une cellule hexagonal, avec R en nombre de pixels
`python3 main.py -o`	Montre seulement les bords des flocons (par exemple pour la découpe laser)
`python3 main.py -t T`	Permet de configurer l'épaisseur des traits
`python3 main.py --text`	Choisi l'interface textuelle sur la console (non-interactive)
`python3 main.py --resolution X Y`	Configure la taille de la fenêtre PyGame à ouvrir
`python3 main.py -p`	Permet d'utiliser les options de configuration de la dernière utilisation

Plus d'options de configurations sont disponibles dans le fichier « settings.py », cependant, ces options sont plus complexes et demandent une affinité avec le programme.

Étapes d'exécution

Lorsqu'on exécute le fichier principal, plusieurs choses se passent de façon automatique. Premièrement, PyGame s'initialise si nécessaire. Puis notre programme analyse les arguments de la line de commande et initialise les options de configuration. Finalement, le processus de simulation commence. On initialise la grille, puis aussi vite que possible, on simule ce qui se passera dans la prochaine étape sur un grille auxiliaires. Une foi toute la simulation

est finit, des statistiques sont collectés, il y a une capture d'écran pour l'animation, et la grille auxiliaires est transférée à la grille primaire et cette première est réinitialisé. Ceci se passe jusqu'à ce que l'utilisateur demande la terminaison du programme, ou une interaction comme une analyse statistique.

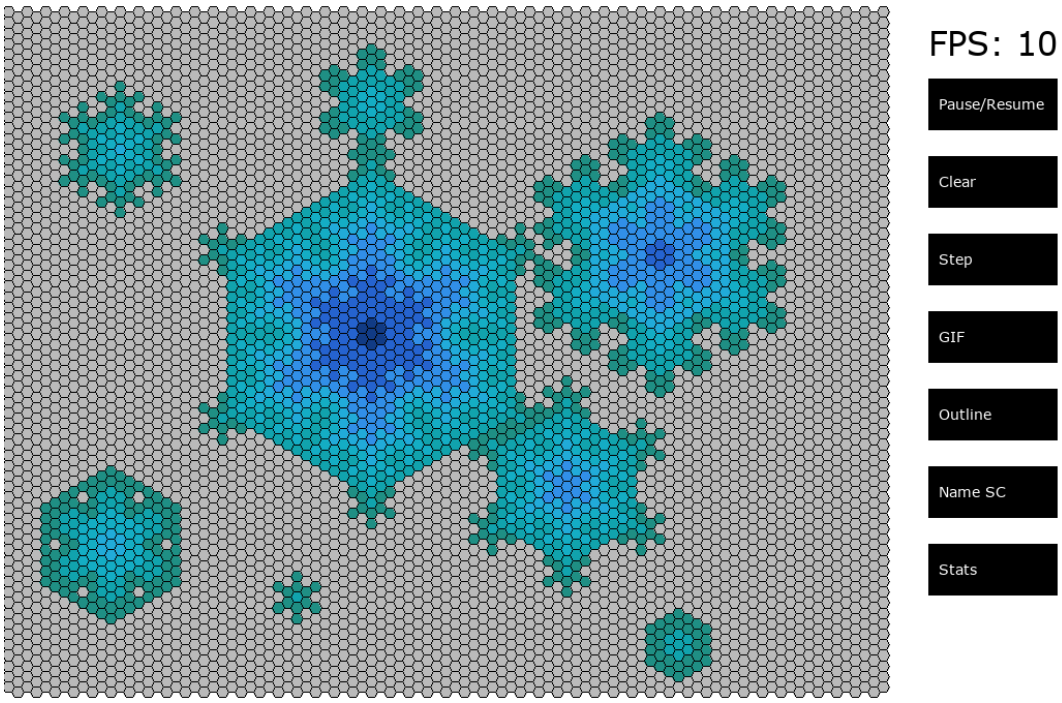
Spécifications Techniques

Le code étant écrit en Python, il est compatible sur la vaste majorité des architectures (x86, x86_64, i686, aarch64, armv7l, ppc64, ppc64le, s390x, arm64…). La dépendance sur PyGame est aussi compatible sur la plupart de ces architectures, bien que cela peut demander plus de travail sur des architectures plus obscures (et sinon, l'option texte et toujours disponibles).

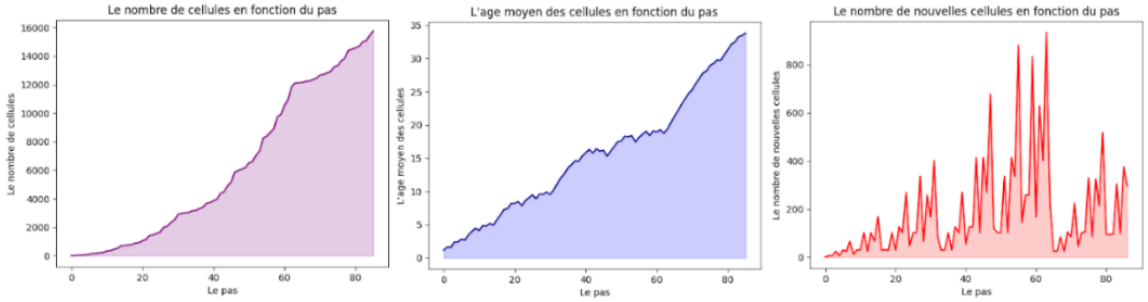
Pour l'interface CLI, nous utilisons le module « *argparse* » de la librairie standard Python et qui vient avec toutes les fonctionnalités dont nous avons besoin. L'interface graphique est créée avec PyGame, qui est la seule librairie qui nous donne la quantité de contrôle dont nous avons besoin tout en gardant le bon montant d'abstraction (comparé à d'autres librairies telle que Qt, plot,ly ou simplement un service web).

Les données de la simulation sont stockées dans la mémoire vive de l'ordinateur sous la forme d'une classe Python et certaines sorties comme les GIFs et schémas de découpe laser sont stockées respectivement dans des fichiers .gif et .png.

La documentation entière et mise à jour peut [se trouve ici](#).

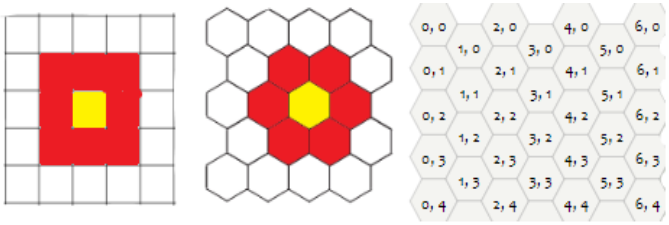


Lien vers une version animée de la simulation exportée automatiquement par le programme



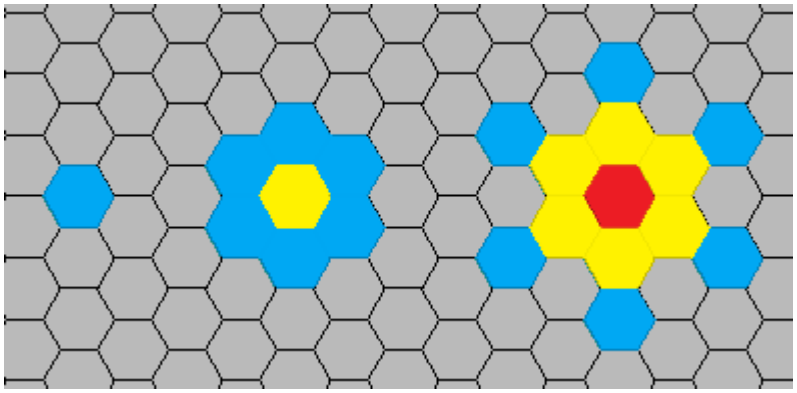
ANNEXE

Figure 1



Une schématisation des voisins (en rouge) d'une cellule donnée (en jaune). A droite il y a une visualisation de notre système de coordonnées pour la grille. Source

Figure 2



Une visualisation des âges des cellules. À chaque étape, les cellules en bleu clair ont un âge d'un, les jaunes ont 2 et puis le rouge à 3 ans. Ce sont les 3 premières étapes de la simulation.

Figure 3

Un flocon découpé avec la machine à laser