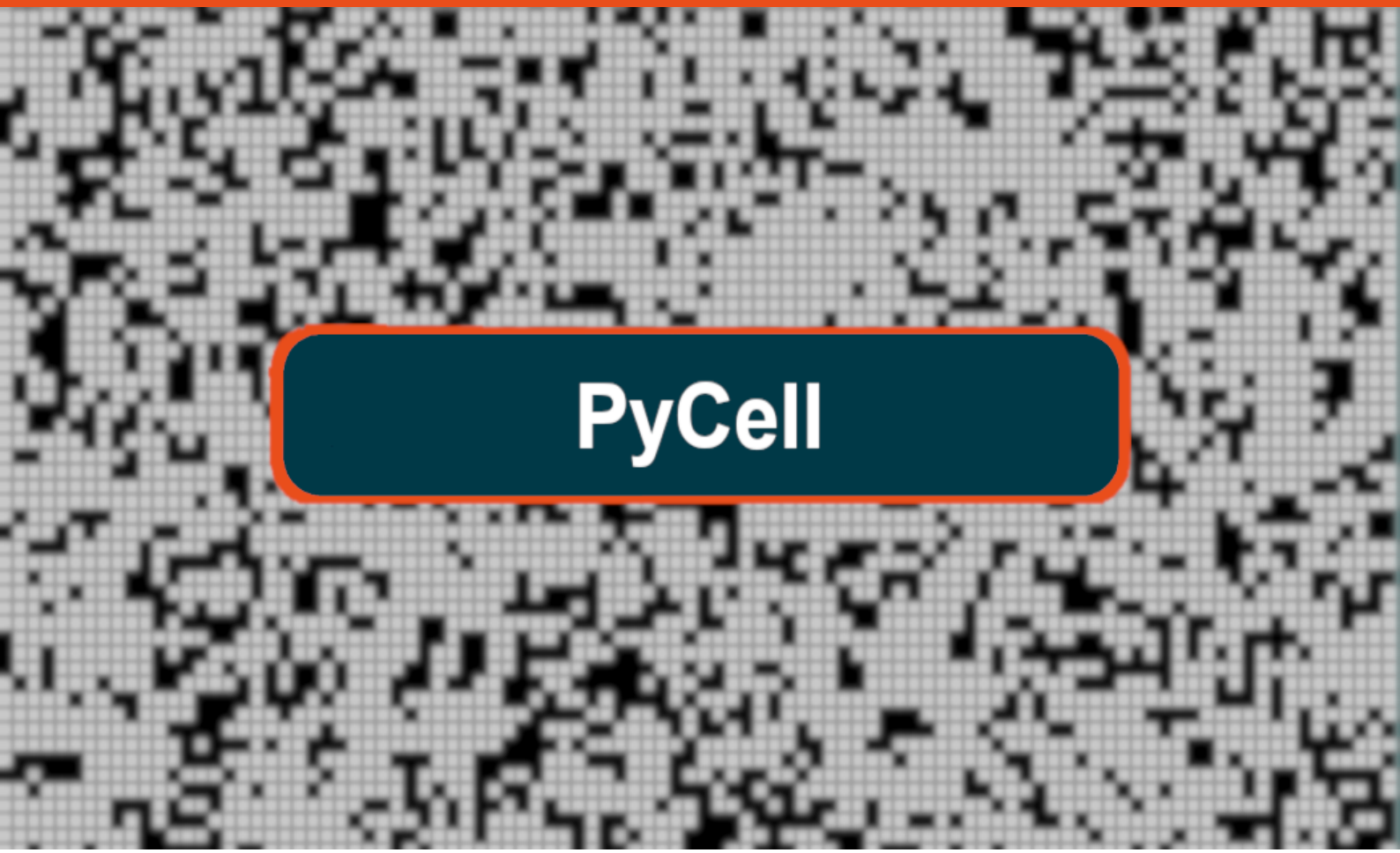
LES TROPHÉES NSI

Édition 2023

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).

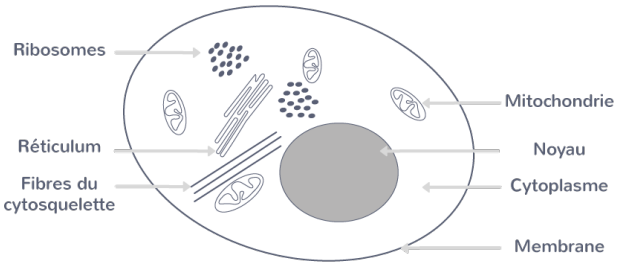
Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

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 : PyCell

> PRÉSENTATION GÉNÉRALE :

Les êtres vivants, même dans leur forme la plus basique : la cellule reste extrêmement complexe en faisant appel à des dizaines si ce n'est des centaines de protéines différentes, à de l'ADN, des mitochondries… Toute cette complexité les rend impossible à simuler dans leur intégralité et avec précision pour pouvoir prédire leurs actions et interactions, même avec les meilleurs ordinateurs a notre disposition. Il est donc inimaginable de pouvoir simuler a plus grande échelle une famille ou même une colonie de ces cellules.



Malgré la croissance très rapide de la puissance de calcul de nos ordinateurs avec un doublement tous les 2 ans du nombre de transistors comme l'avait prédit Gordon Earl Moore depuis plus de 50 ans, la puissance de calcul nécessaire pour parfaitement simuler une cellule reste bien loin de notre porté.

Nous nous sommes donc posé la question : **Quelles solutions avons-nous pour simuler et donc mieux comprendre les cellules et leurs colonies ?**

Pour trouver une réponse à cette question nous pouvons essayer de simplifier les propriétés d'une cellule en affirmant qu'elle est soit morte soit vivante et qu'elle doit avoir la capacité de se reproduire seul.Nous pouvons rajouter que son état dépend de son entourage.

John Von Neumann parvient à créer un système qui s'autoreproduit au milieu du XXe siècle mais les règles restent assez complexes. C'est John Horton Conway qui a réussi à simplifier ces règles en créant en 1970 le jeu de la vie. Les règles sont simples : une cellule naît si elle est entourée de 3 autres cellules vivantes, garde son état si elle a 2 voisins vivants et meurt sinon. Malgré les règles simples, la puissance de calcul de l'époque était trop faible pour pouvoir simuler ce jeu, forçant Conway à jouer sur un plateau de Go. Heureusement, aujourd'hui un simple ordinateur peut faire ce genre de simulation.

Ces quelques règles très simples permettent la création de structures beaucoup plus complexe comme des vaisseaux qui peuvent se déplacer sur la grille, des canons qui peuvent créer des vaisseaux, des oscillateurs qui reprennent leur forme initiale après plusieurs cycles ou même des structures stables qui ne changent pas d'un cycle à l'autre.

> ORGANISATION DU TRAVAIL :

Dimitri Gallet : Création de l'interface graphique a l'aide de PyGame
Gabriel Keene : Rédaction du code de la simulation et de la pose de structures
Mattéo Cid-Saliou : Rédaction de la documentation, des docstrings et simplification des commentaires
Maxime Blanchard : Réalisation de la vidéo et rédaction des différents documents

Les tâches ont été réparties au début du projet ce qui nous a permis de gagner beaucoup de temps. Cependant, la répartition des tâches n'était pas si stricte : dès que quelqu'un avait besoin d'aide tout le monde pouvais intervenir et l'aider.Si bien que chacun a un petit peu participé à tout. Nous n'avons pas eu de réunion formelle pour discuter du projet car chacun avançait à son rythme et que des réunions, en plus de prendre du temps auraient forcé chacun a faire une sorte de compte-rendu hebdomadaire de plus, il est très difficile se trouve un horaire commun en raison de nos différents emplois du temps.Au contraire nous avons préféré utiliser une messagerie instantanée qui permet à chacun d'indiquer ses avancés et de demander de l'aide si nécessaire.

La rédaction d'un pseudo-code avant python nous a permis d'éviter la majorité des bugs et la plupart de ceux restant ont été réglé par une rapide séance de débogage. Pour les quelques bugs plus coriaces nous pouvions demander de l'aide au groupe et le problème était rapidement résolus.

Le code était stocké sur un serveur Git avant d'être mis sur Github une fois le développement fini

LES ÉTAPES DU PROJET :

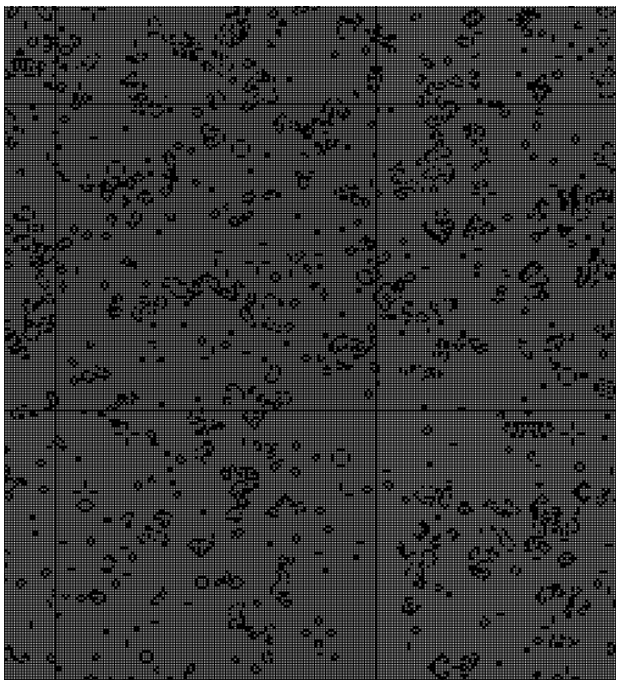
L'idée de ce projet nous est venus en cours d'enseignement SVT alors que nous abordions la cellule. Nous nous sommes demandé s'il était possible de simuler des cellules pour pouvoir étudier leur comportement à différentes échelles.

Quelques jours après nous avons découvert les trophées NSI et nous nous sommes dit que le problème de la cellule serait une bonne piste à explorer. Nous avons donc formé l'équipe et nous nous sommes mis au travail.

Le projet a progressé à différentes vitesses en parallèle en fonction des disponibilités de chacun avec des mises en commun quand nécessaire

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Nous avons surmonté plusieurs difficultés au cours de ce projet. La principale fut l'apparition de motif sur la grille lorsque l'on dézoomait. Nous avons pris du temps avant de comprendre qu'il s'agissait de l'effet moiré. Pour résoudre ce problème nous avons essayé sans succès l'anticrénelage avant de devoir supprimer le quadrillage quand les lignes devenaient trop nombreuses. Nous avons également fait face à des pertes conséquentes de performances avec des grandes grilles. Ce problème a été résolu en réduisant le nombre de case a vérifier en ne vérifiant que les voisines des cellules vivantes.

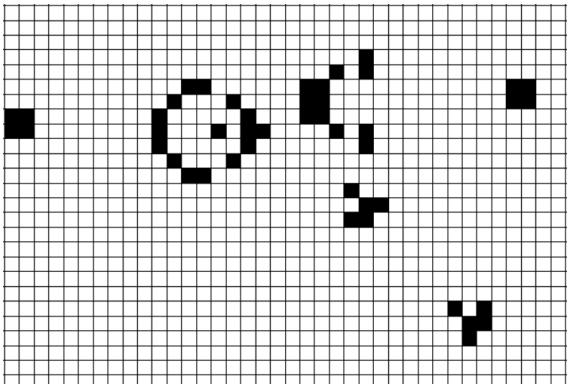


Effet Moiré visible(zoomer sur l'image si il n'est pas visible)

À l'heure actuelle nous sommes contents d'avoir pu faire ces fonctionnalités :

-Le jeu de la vie: La gestion du jeu en lui-même avec les interactions entre cellules et la possibilité de modifier la grille en cliquant dessus.C'est ce que nous avons fait en premier. Pour tester cette fonctionnalité nous avons pu créer des structures complexes qui font appel à toutes les propriétés comme des canons et vérifier que leur comportement empirique était en accord avec leur comportement théorique ce qui était le cas

-L'adaptation de la taille de la grille: Cette fonctionnalité permet de grâce à la molette de la souris ou un bouton virtuel de changer très rapidement la taille de la grille simulée et donc de limiter les pertes de performances inutiles. Pour vérifier cette fonctionnalité nous avons simplement testé le logiciel pour se rendre compte qu'il n'y avait aucun problème.



Un canon a planeur dans notre simulation

-Le changement des règles : Grace à cette fonctionnalité l'utilisateur peut désormais changer les règles du jeu de la vie à sa guise pour pouvoir simuler d'autres automates cellulaires et découvrir d'autres comportements, d'autres structures… Pour vérifier que cela fonctionnait correctement nous avons testé tous les boutons et observé si les changements étaient bien appliqués dans les tableaux stockant le nombre de case pour qu'une cellule naisse ou meure.

-L'affichage de l'historique du nombre de cellules vivantes : Cette fonctionnalité permet de voir très rapidement s'il y a un organisme qui s'autoreproduit, si le système est stable ou bien si la population diminue

-L'optimisation des calculs : dans les premières versions du projet, il fallait vérifier chaque cellule pour savoir si elle devait vivre ou mourir. Nous avons pu observer qu'apres quelque itération le nombre totale de cellule diminue très souvent en créant des zones vides. Les cellules qui ne sont pas voisines de cellule vivante n'ont aucune chance de changer d'état(sauf si 0 voisins font naître une cellule, ainsi il faut tous vérifier).Ainsi nous créons un ensemble de cellules voisines de cellule vivante et donc susceptible de changer d'état et nous ne vérifions pas les autres cellules ce qui permet un énorme gain de performance (entre 2 et 4 fois moins de cellule à vérifier après une dizaine d'étapes)

-La création d'images animées : cette fonctionnalité permet de créer des images animées de la grille d'un simple clique afin de pouvoir ensuite les diffuser par le(s) canaux souhaités.

> OUVERTURE :

Bien que nous soyons très fiers de notre projet nous sommes conscients qu'il est perfectible et que des fonctionnalités peuvent être ajoutées. Nous avons donc eu quelques idées que nous allons essayer d'implémenter dans le futur. Notamment :

-La création d'autres structures prédéfinies, en fonction des règles si elles ont été changé car les structures actuelles ne fonctionnent pas dans toutes les configurations possibles.

-**La possibilité de compter les cellules qui « sortent » de l'écran** car pour l'instant la simulation fait comme si elle cessait d'exister ce qui peut fausser les chiffres du nombre de cellules vivantes

- Modifier l'interface graphique pour pouvoir contrôler avec des boutons virtuels le temps entre chaque génération, si l'on souhaite cacher ou non les informations et si l'on souhaite mettre en pause.

- Il pourrait être intéressant de **changer la couleur des cellules en fonction de si elles viennent de naître ou de mourir** afin d'observer les évolutions d'une structure.

Pour la diffusion de ce projet nous avons commencé par en parler et le montrer à nos proches (familles, amis…) qui nous ont donné des retours très constructifs car ils ne faisaient pas partie du projet et avaient donc une autre vision. Ces retours nous ont permis de faire de nombreuses modifications et amélioration. Pour viser un public plus large nous avons commencé une chaîne Youtube sur laquelle nous publions des vidéos montrant différentes structures qui évoluent. Cela nous permet d'atteindre un public tout autre et de lui faire découvrir notre projet. Y compris un public plus jeune pour l'intéresser aux sciences numériques et peut être lui donner envie d'en apprendre plus.

DOCUMENTATION

Pour installer ce projet il faut exécuter ces commandes qui permettent : de cloner notre projet, de se déplacer dans le dossier, d'installer les bibliothèques requises puis enfin, de l'exécuter.

```
$ git clone https://github.com/GabrieKeene/PyCell
$ cd PyCell
$ pip install -r requirements.txt
$ py main.py
```

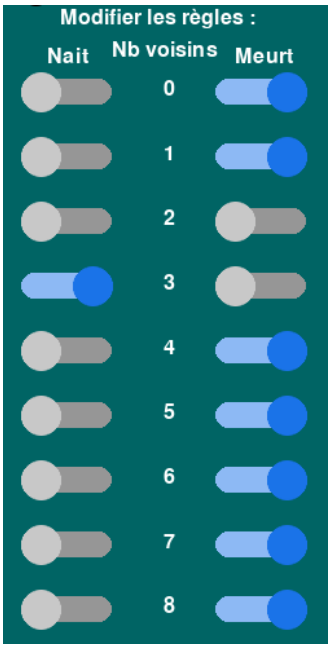
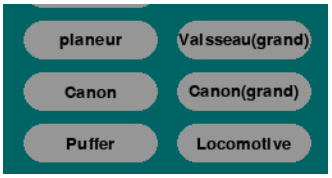
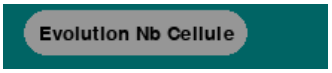
Ce programme fait appel à plusieurs bibliothèques :

-PIL pour l'exportation des images animées
-Pygames pour gérer l'affichage et les entrées utilisateurs
-Pygames-widgets pour afficher les boutons et le slider
-Matplotlib pour afficher le graphique montrant le nombre de cellules en vie en fonction du temps

Par souci de simplicité et parce que la taille des données nous le permettait, tout est stocké dans le fichier main .py. Lors des exports d'images, les formats GIF est utilisé et les images animées sont stockées dans le dossier image.

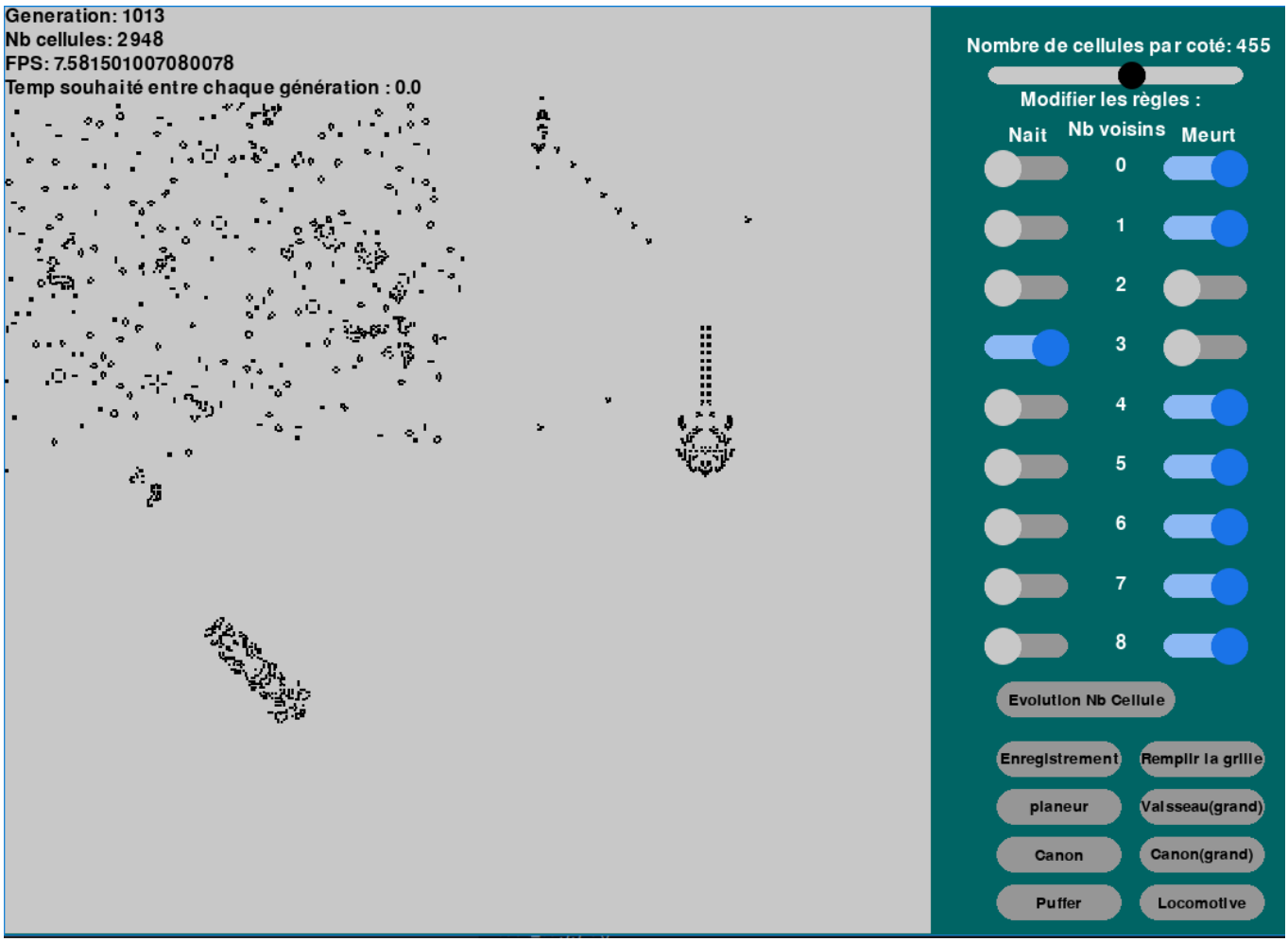
Une fois le programme installé et démarré l'intégralité des contrôles se fait au clavier et/ou à la souris. Les actions possibles sont :

Nom de la fonctionnalité	Moyen d'action	Explication	Capture d'ecran
Changer la taille de la zone simulé	Molette de la souris / Slider en haut a droite		

	de l'écran		
Modifier les régles du jeu	Boutons sur la droite de l'écran	Un bouton est bleu quand il est activé. La colonne de droite indique si une cellule avec le nombre de voisins indiqué naît, celle de gauche si elle meurt. Si aucun des deux boutons n'est activé la cellule garde son état	
Créer un enregistrement	Bouton a droite de l'écran	Démarre et termine un enregistrement stocké dans le dossier image. L'enregistrement s'arrête seul après 500 images	
Poser une structure prédéfinie	Bouton en bas a droite de l'écran puis clique sur la grille	Permet de disposer sur la grille une ou plusieurs des six structures prédéfinies	
Voir l'historique du nombre de cellule	Bouton a droite de l'écran	Affiche l'historique du nombre de cellules vivantes par génération	
Afficher/cacher les informations	Touche H du clavier	Affiche ou cache le texte en haut a gauche du clavier	
Remplir la grille	Touche R du clavier ou bouton a droite de l'ecran	Remplis la grille d'un motif aléatoire de densité 1/6 en moyenne	
Changer le temp entre les générations	Touche fleche haut et fleche bas du clavier	Change le temps minimum entre chaque génération. Le temps est affiché en	

		haut a gauche en seconde si le texte n'est pas caché	
Mettre en pause la simulation	Touche P du clavier	Met en pause le jeu jusqu'à ce que la touche soit a nouveau pressé	

Le programme peux en théorie s'exécuter sur Windows, mac et Linux mais il n'a été testé que sur Windows et Linux (Ubuntu).



Capture d'écran de la simulation,: on peux observer un vaisseau en bas a gauche, un puffer qui laisse une trainé a droite, un canon en haut et diverses structures dans le quart en haut a gauche