Jeu de sauvetage par drones

HAIDA Ilyane

GARIAZZO Ruben

Rapport de Projet

ENGINEERING SCHOOL

Sommaire du rapport

Introduction

Notre projet sur python vise à implémenter un jeu tour par tour sur une grille 12×12, où des drones doivent sauver des survivants tout en évitant des tempêtes et des obstacles. Le programme respecte les contraintes de modélisation (dictionnaires Python pour les entités), lit sa configuration depuis un fichier JSON, génère aléatoirement les positions initiales, et produit un journal des événements ainsi qu'un score final. Ce rapport décrit la conception, les données et les fonctions du programme.

Conception du programme

Structure générale

Le programme est structuré en cinq grandes parties :

La première partie consiste en le chargement de la configuration, ce qui comprend la lecture du fichier de configuration programmé en JSON (config.json) avec json.load(). Ce fichier contient tous les paramètres du jeu : taille de la grille, nombres de drones, tempêtes, survivants, bâtiments, batterie initiale, recharge, etc. Grâce à cette approche, les réglages sont modifiables sans toucher au code.

La deuxième partie est la génération aléatoire des positions, soit le placement des bâtiments qui sont des obstacles récurrents, l'hôpital, les survivants, les 4 tempêtes et les 6 drones, tout en empêchant la superposition en enregistrant les positions déjà utilisées dans une liste nommée occ. Chaque entité est créée avec un dictionnaire contenant son identifiant, sa position et d'autres attributs spécifiques (batterie, état, charge pour les drones). Cette génération garantit que chaque case ne contient qu'un seul élément au départ.

La troisième partie regroupe les fonctions d'affichage. La fonction construire_grille() construit une grille 2D (liste de listes) à partir des positions actuelles des entités. La fonction afficher_grille() affiche cette grille à l'écran avec les colonnes identifiées par des lettres (A, B, C...) et les lignes par des chiffres, pour une lecture facile par le joueur.

Une fonction afficher_drones() affiche l'état détaillé de chaque drone (position, batterie, état, survivant transporté).

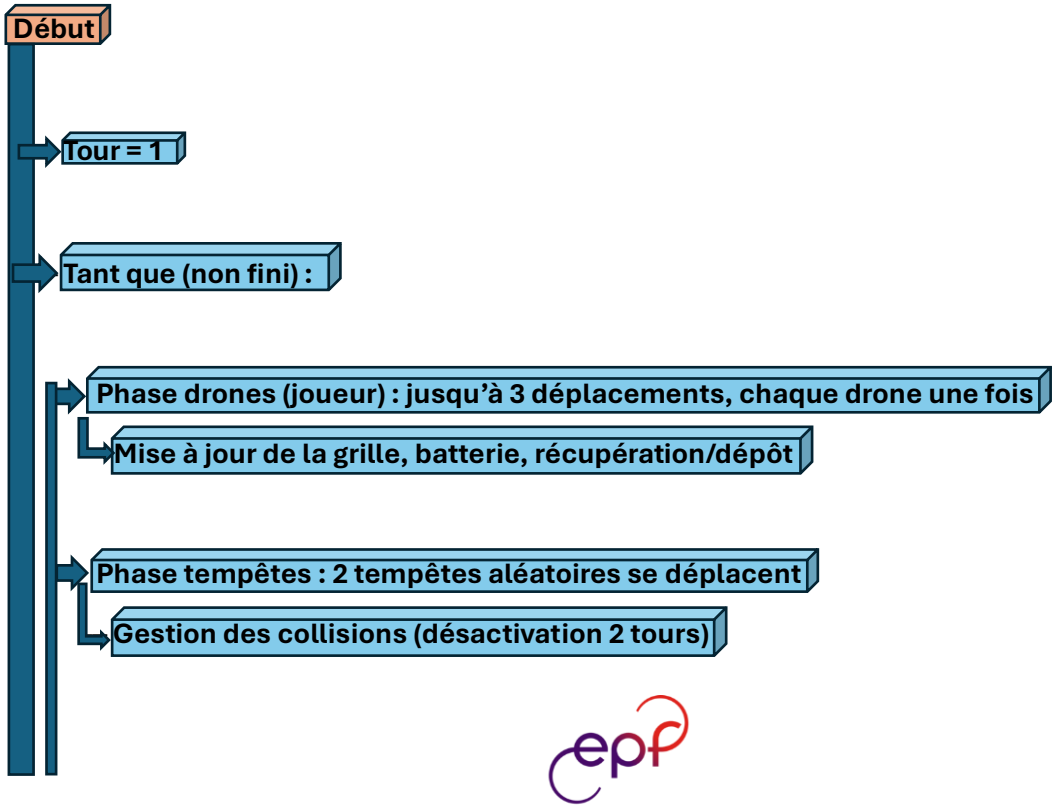
La quatrième partie est le cœur du jeu : les fonctions de déplacement et de gestion des règles. On y trouve deplacement_valide() qui vérifie si une case est accessible (dans la grille et pas un bâtiment). deplacer_drone() gère le mouvement d'un drone : calcul du coût énergétique (1 par case, +2 si le drone prend un survivant), mise à jour de la batterie, récupération d'un survivant, dépôt à l'hôpital (qui augmente le score), et recharge si le drone termine sur l'hôpital. deplacer_tempete() déplace une tempête d'une case dans une direction aléatoire valide, et si une tempête atterrit sur un drone actif, celui-ci est désactivé pour deux tours. update_desactivation() décrémente chaque tour le compteur de désactivation des drones touchés et les réactive lorsque le compteur atteint zéro. Enfin, recharger_drones() est appelée à la fin de chaque tour pour ajouter 3 de batterie à tous les drones stationnés sur l'hôpital.

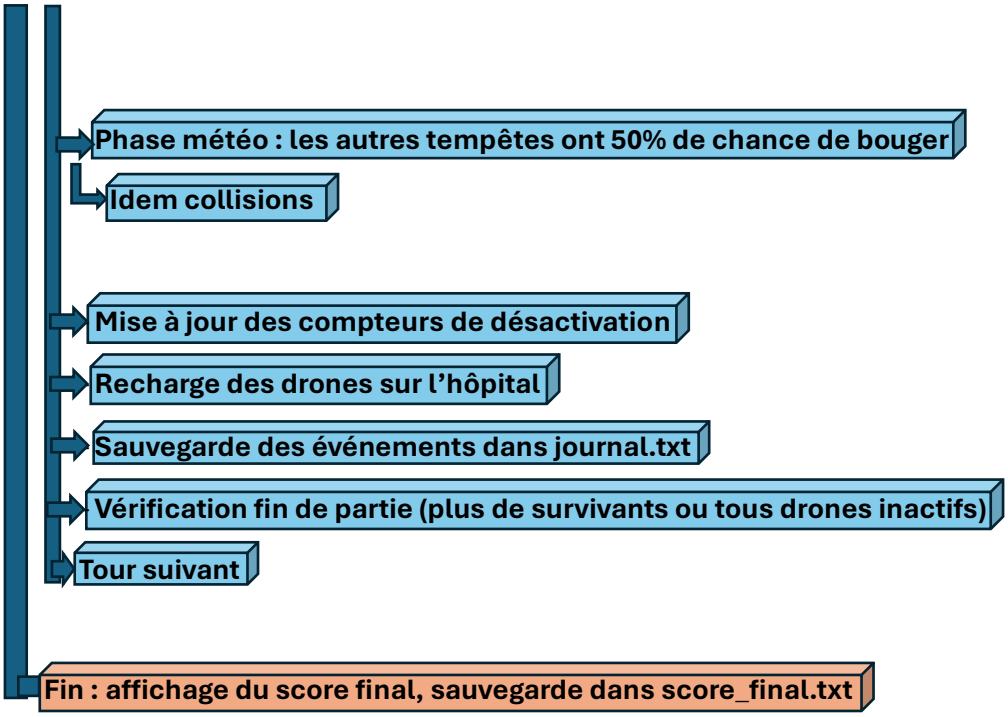
La cinquième partie est la boucle principale du jeu. Elle organise le déroulement des tours :

- Affichage de la grille, du score et des drones.
- Phase drones : le joueur peut déplacer jusqu'à trois drones différents (un seul mouvement par drone par tour).
- Phase tempêtes : deux tempêtes sont choisies aléatoirement et se déplacent.
- Phase météo : les tempêtes restantes ont 50% de chances de se déplacer.
- Mise à jour des désactivations, recharge des drones sur l'hôpital.
- Sauvegarde des événements du tour dans un fichier journal (journal.txt).
- Vérification des conditions de fin de partie (plus de survivants ou tous les drones inactifs).
- Incrémentation du numéro de tour et répétition.

En fin de partie, le score final est écrit dans un fichier score_final.txt.

Schéma de boucle de jeu





Variables, fonctions et fonctionnalités

Variables principales

Nom de la variable	Type	Rôle / utilisation
cfg	dictionnaire	Paramètres lus depuis config.json (taille, nombres, règles)
T	entier	Taille de la grille (12)
occ	liste	Liste des positions déjà occupées (empêche les superpositions)
batiments	liste de tuples	Positions des obstacles fixes (symbole B)
hopital	tuple (r,c)	Position de l'hôpital (symbole H)
survivants	liste de dict	Chaque dict : {"id":int, "pos":(r,c)}
tempetes	liste de dict	Chaque dict : {"id":int, "pos":(r,c)}
drones	liste de dict	Chaque dict : {"id":int, "pos":(r,c), "batt":int, "etat":str, "charge":int/None, "desact":int}
score	entier	Nombre de survivants sauvés

Nom de la variable	Type	Rôle / utilisation
tour	entier	Numéro du tour actuel
journal	liste de chaînes	Événements du tour avant écriture dans journal.txt
deplaces	liste d'entiers	IDs des drones déjà déplacés dans le tour courant
indices	Liste d'entiers	Stocke les deux indices des tempêtes déplacées lors de la phase tempêtes.
tous_inactifs	booléen	Variable utilisée pour vérifier si tous les drones sont inactifs (remplace la fonction all)

Liste des fonctions

Fonction	Entrées	Sorties	Rôle
positions_aleatoires(n, excl, occ)	n : entier excl : liste occ : liste	liste de tuples (r,c)	Génère n positions uniques, hors excl et occ
construire_grille()	aucun	liste de listes (grille 2D)	Construit la grille à partir des entités
afficher_grille()	aucun	None (affichage)	Affiche la grille avec colonnes lettres / lignes chiffres
afficher_drones()	aucun	None	Affiche l'état de chaque drone (batterie, état, charge)
recharger_drones(journal, tour)	journal : liste tour : entier	None	Recharge (+3) les drones sur l'hôpital
deplacement_valide(drone, dest)	dest : tuple (r,c)	booléen	Elle vérifie que la destination est dans la grille, que ce n'est pas un bâtiment, que le déplacement ne dépasse pas une case, et que la case n'est pas déjà occupée par un autre drone

Fonction	Entrées	Sorties	Rôle
deplacer_drone(d, dest, score, journal, tour)	d : dict drone dest : tuple score : int journal : liste tour : int	(score, journal)	Déplace un drone : coût batterie, récupération, dépôt, recharge
deplacer_tempete(t, journal, tour)	t : dict tempête journal : liste tour : int	journal	Déplace une tempête aléatoirement, gère collisions
update_desactivation(journal, tour)	journal : liste tour : int	None	Décrémente les compteurs de désactivation, réactive les drones
positions_aleatoires(n, excl, occ)	n : entier excl : liste occ : liste	liste de tuples (r,c)	Génère n positions uniques, hors excl et occ
construire_grille()	aucun	liste de listes (grille 2D)	Construit la grille à partir des entités
afficher_grille()	aucun	None (affichage)	Affiche la grille avec colonnes lettres / lignes chiffres

Récapitulatf des fonctionnalités

Fonctionnalité	État	Remarque
Grille 12×12 avec symboles (D, S, T, B, H, .)	Correct	Colonnes en lettres, lignes en chiffres
6 drones, 4 tempêtes, 10 survivants	Correct	Génération aléatoire sans superposition
Déplacements drones (8 directions, 1 case)	Correct	Vérification des obstacles
Batterie : 1 par déplacement	Correct	Coût de base
Prendre un survivant coûte 2 supplémentaires	Correct	Ajout du coût lors de la récupération

Batterie à 0 rend le drone immobile	Correct	État passe à « immobile »
Recharge +3 par tour sur l'hôpital (max batterie)	Correct	Appliquée à la fin de chaque tour
Désactivation 2 tours si tempête touche un drone	Correct	Collision détectée après déplacement
Phase drones : max 3 déplacements, un seul par drone	Correct	Liste deplaces pour éviter doublons
Phase tempêtes : 2 tempêtes aléatoires se déplacent	Correct	Choix de deux indices distincts
Phase météo : autres tempêtes 50% de chance	Correct	Utilisation de random.random() < 0.5
Lecture configuration depuis JSON	Correct	open() + json.load()
Génération aléatoire de toutes les positions	Correct	Aucune coordonnée fixe
Déplacement limité à une case (8 directions, pas de saut)	Correct	Le drone ne peut se déplacer que d'une case adjacente
Interdiction de superposition des drones	Correct	Un drone ne peut pas se déplacer sur une case occupée par un autre drone
Fonctionnalité	**État**	**Remarque**

Conclusion

Ce projet a permis de mettre en pratique l'ensemble des concepts fondamentaux de nos cours de Python, à travers des structures de données (listes, dictionnaires), fonctions, gestion de fichiers, génération aléatoire, et logique de jeu tour par tour.

Les principales difficultés rencontrées ont concerné la gestion des collisions entre tempêtes et drones ainsi que le compteur de désactivation. Elles ont été résolues par l'ajout d'une variable desact et d'une fonction de mise à jour appelée à chaque tour.

Les limites actuelles tiennent à l'absence de bonus (météo dynamique, stations de recharge multiples, spécialisation des drones). L'interface en ligne de commande reste simple mais suffisante pour démontrer le fonctionnement du jeu. À l'avenir, on pourrait enrichir l'expérience utilisateur avec une interface graphique ou ajouter des niveaux de difficulté variables.

C'était la première fois que nous écrivions un programme long et complexe, en respectant un cahier des charges à la livraison des fichiers attendus (configuration, journal, score). Il a également conforté l'importance de la rigueur dans la gestion des données et la séparation des responsabilités entre fonctions.