ESN Wars

Résumé

Dans ce projet de la SAE1.01, vous allez développer un jeu de stratégie en tour par tour pour 2 à 4 joueurs nommé ESN Wars.

Une ESN est une *Entreprise de Services du Numérique*, anciennement appelée SS2I. Dans la SAE1.02, vous implémenterez une IA pour ce jeu.

Vue d'ensemble du jeu

Dans une grande ville technologique du futur proche, où le travail en distanciel a été interdit, les *freelance coders* (ou coders) rivalisent de concurrence pour achever des missions informatiques au sein de diverses entreprises, et gagner assez de bitcoins pour s'offrir la PS13 qui vaut la bagatelle de 5000฿ (mais elle est top).

Objectif du jeu : le premier joueur qui finit son tour avec 5000฿ a gagné. Si au bout de 1000 tours personne n'a atteint l'objectif, le joueur avec le plus d'argent est déclaré vainqueur, et la partie nulle en cas d'égalité.

Présentation des règles

La carte

Le jeu se joue sur une grille rectangulaire 21*21 , dont les cases sont repérées par des coordonnées entières(i,j) de 0 à 20.

La case centrale (10,10) contient le Job Center (JC). C'est un endroit où les coders peuvent se reposer (boire un redbull et jouer à la PS12), ainsi que se former ou acquérir de nouveaux logiciels.

Les coders

Chaque joueur déplace un coder qui est un personnage occupant une case de la carte. Deux coders ne peuvent jamais être sur la même case, hormis au Job Center. Chaque coder démarre la partie sur le Job Center. Chaque coder dispose de 4 caractéristiques :

- son **coding level** (CL) qui commence à 1 (maximum : 10). Le niveau permet d'effectuer des missions plus rapidement.
- son **énergie max** (EM), qui commence à 1 (maximum : 10). C'est le niveau que l'énergie ne pourra jamais dépasser à chaque moment.
- son **énergie** (E), qui commence à 1 et peut aller jusqu'au niveau d'énergie max. Un coder ayant 0 énergie ne peut plus réaliser de mission, il doit retourner au JC afin de se reposer.

- sa richesse (R), qui commence à 0฿ et n'a pas de maximum, mais dès que 5000฿ est atteint la partie va se terminer. La richesse ne peut pas être négative.

Les missions

Sur la carte certaines cases contiennent des missions, que les joueurs peuvent voir vec leurs caractéristiques. Chaque mission est caractérisée par :

- son **starting workload** (SW), qui est la charge totale du travail à effectuer pour terminer la mission
- son **remaining workload** (RW), qui est charge de travail restante à effectuer
- sa **difficulté** (D), qui détermine l'énergie nécessaire à un coder pour avancer dans la mission.

Pour gagner de l'argent, un coder doit réaliser des missions. Pour cela il doit se rendre sur les cases mission, et réalisera une partie de la charge de travail. Ceci lui coutera de l'energie. Le codeur qui achève une mission (amène à 0 la remaining workload) gagne de l'argent.

Règles précises

Mise en place

Au début de la partie les coders sont tous positionnés sur le coding center, au centre de la grille. Les missions sont distribuées aléatoirement ou selon un niveau prévu à l'avance.

Niveau d'information

A tout moment de la partie, toutes les informations (positions des coders, caractéristiques, positions des missions, caractéristiques, etc) ainsi que toute la liste des coups joués par le passés sont accessibles par les joueurs pour prendre leurs décisions, rien n'est caché.

Ordre du tour

Les 4 joueurs jouent leur tour de façon cyclique, du 1 au 4 puis on recommence au 1 et ainsi de suite, comme dans la plupart des jeux de société classiques. L'ordre peut être tiré au hasard ou décidé à l'avance.

Actions

A chacun de ses tours, un coder peut réaliser une et une seule action. Le codeur peut également ne pas faire d'action (passer). Ces actions peuvent être un déplacement ou bien un upgrade.

Déplacements

Il y a 4 déplacements possibles dans chacune des 4 directions de base (Haut,Bas,Gauche,Droite), sans sortir de la carte. On ne peut pas se déplacer sur une case contenant un autre coder sauf sur le JC. Se déplacer est toujours possible.

Upgrade

Si un coder se trouve sur le JC, il peut réaliser une des actions suivantes qui sont des upgrades :

- augmenter de 1 son énergie max ;
- ou bien augmenter de 1 son coding level.

Dans les deux cas le coût en dollar est égal au carré du niveau désiré, fois 10\฿. Ces actions sont possibles seulement si le coder dispose de l'argent nécessaire.

Exemple : Il faut 40฿ pour passer de 1 à 2, 90฿ pour passer de 2 à 3 et 160฿ pour passer de 3 à 4.

Avancer une mission

Si un coder termine son tour sur une case contenant une mission ayant une RW non nulle, alors le coder travaille à la mission (ce n'est pas compté comme une action).

- le coder perd un nombre de points d'energie égal à la difficulté de la mission. Si son énergie devient négative, elle est ramenée à 0. Il est donc possible de travailler sur une mission de difficulté supérieure à son coding level, mais on y perdra toute son énergie.
- le RW de la mission diminue du CL du coder

Terminer une mission

Si en avançant une mission, le coder amène la RW à 0 (ou moins), la mission est réalisée et le coder gagne un revenu égal au produit SW * D.

*Exemple : on a une mission de SW=RW=4, D=2, et un coder de CL=3 et E=5. Au tour 1, le coder se déplace sur la mission. Le RW passe à 4-3=1, le coder passe à E=5-2=3. Au tour 2, le RW passe à 1-3=-2, ramené à 0, le coder passe à E=3-2=1. La mission est terminée, le coder gagne 4 * 2 = 8฿.*

Attention, seul le coder qui termine la mission gagne le revenu. Si un codeur doit retourner au JC car il n'a plus d'énergie en travaillant à une mission (voir ci-dessous), un autre coder peut venir travailler à la mission.

Reboot des missions

Quand une mission vient d'être effectuée, elle devient indisponible pour un nombre de tour égal à sa difficulté * 10 puis reprend ses valeurs de départ.

Resting Place

Si un coder termine son tour sur le job center, il regagne toute son énergie (ce n'est pas une action).

End Game

Lorsqu'un coder, à la fin de son tour, dispose de 5000฿, alors le match se termine.

Objectifs d'implémentation

L'objectif final idéal de votre projet est d'avoir une version graphique du jeu ergonomique qui implémente de façon exacte toutes les règles, où un joueur humain peut jouer contre un à trois joueurs IA. Des objectifs plus raisonnables sont détaillés par les niveaux suivants. Essayez de complètement réaliser un niveau avant de passer au suivant.

Niveau 0

Version jouable en ligne de commande, où les joueurs entrent leurs actions dans le terminal. On peut envisager au départ une version avec un seul joueur. Les règles doivent être respectées. L'affichage de la carte se fait dans le terminal avec un système de lettres. La map est toujours la même et elle est décrite dans le code.

Niveau 1

On peut jouer jusqu'à 4 joueurs, et un système de lecture de fichiers permet de charger le niveau voulu.

Niveau 2

Au niveau 2 vous fournissez une interface graphique pour le jeu.

Niveau 3

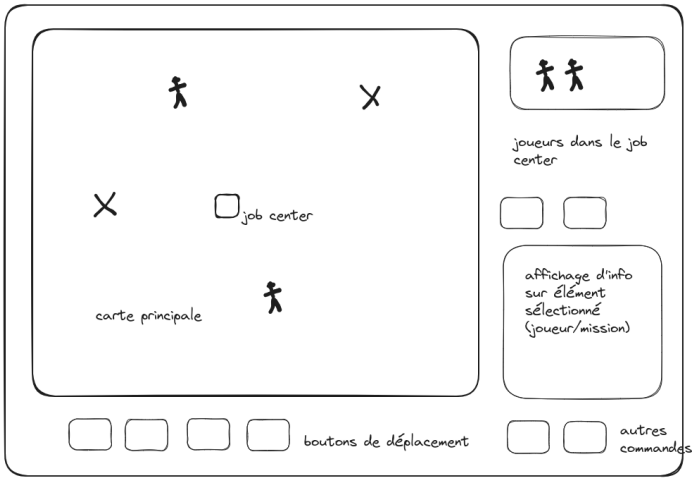
Idem, mais vous fournissez des IA simples qui permettent de jouer contre l'ordinateur. Le but de cette SAE n'est pas de réaliser des IA performantes. Vous pouvez ajouter des fonctionnalités : système de sauvegarde, annulation d'un coup, etc.

Recommandations

La partie graphique

- Le moteur de jeu (la partie logique et données) doit être séparé de la partie graphique. Aussi, il faut réaliser le niveau 1 et valider toutes les règles avant de passer au niveau 3. Ceci n'empêche pas de développer l'interface en parrallèle avant de raccorder les deux.
- Pensez pratique, idéalement on doit pouvoir faire tous ses choix à la souris. Pensez aussi que le jour du test on doit arriver rapidement au jeu sans entrer des informations au clavier.
- N'utilisez pas d'autre bibliothèque graphique que tkiteasy ou tkinter. Le code doit tourner directement sur les machines de l'IUT.

Voici une possibilité de présentation de l'interface :



Vous pouvez également réfléchir à mettre les informations directement sur la carte, ou utiliser un code couleur pour repérer les missions plus ou moins faciles.

Le style du code

- Utilisez des variables globales constantes pour les dimensions de la fenêtre et autres paramètres susceptibles de varier (comme la durée de la partie, etc.)
- Votre code doit être découpé en fonctions, et hormis les constantes indiquées ci-dessus ne doit pas utiliser de variable globale. Si du code est dupliqué, c'est qu'il fallait utiliser une fonction avec un paramètre pour indiquer ce qui change. Idéalement, une fonction fait moins de 15 lignes.
- Vos variables et fonctions doivent avoir des noms intelligents qui permettent de comprendre.
- Commentez les grandes étapes du code sans le paraphraser, et mettez toutes les spécifications (docstring) des fonctions.
- Fournissez le fichier des tests effectués.

Votre travail

- Le travail en binôme est obligatoire. Le travail en projet est une occasion de beaucoup progresser, aussi ne laissez pas l'autre tout faire ! Vous seriez également pénalisés lors de l'évaluation où nous nous assurons que **chacun des deux maîtrise l'ensemble du projet** par des questions.
- Vous suivrez la progression par étapes proposée dans ce sujet afin que votre programme soir bien structuré. La note maximale sera donnée aux projets qui atteignent le dernier niveau et qui sont clairs, bien écrits, stables ; mais nous préférons un programme bien écrit, stable, et qui va moins loin dans les niveaux, plutôt qu'un programme très sophistiqué mais mal conçu, pas clair et et plein de bugs. La note dépend largement autant de votre maîtrise et la clarté/conception du code que de l'application en elle-même.
- S'aider d'un groupe à l'autre est possible, recopier un code tel quel est interdit. Nous avons des moyens informatiques de comparaison des codes (JPlag, entre autres). Le code de votre binôme doit être le vôtre, et chacun.e des membres du binôme doit être capable de l'expliquer intégralement, que ce soit « votre partie » ou non, comme déjà dit (mais on insiste).

Les échéances

- Vous déposez votre projet à l'endroit qui sera indiqué avant le **SAMEDI 16 DECEMBRE à 23h59**. Il doit être accompagné d'un README de quelques pages permettant de le prendre en main. Ce README est en anglais, voir avec votre enseignant d'anglais pour les consignes précises.
- Des soutenances de projet auront lieu dans la semaine qui suit. Environ 10 à 15 minutes par binômes, nous regardons les projets, le code des projets, et votre maîtrise de ce code.

Les compétences qui seront évaluées sur ce projet seront les suivantes (il peut y avoir des changements mineurs) :

- Respect du cahier des charges de dépôt (deadline, README, archive nettoyée et au bon format, programme qui fonctionne directement sans erreurs et sans rien installer de nouveau sur les machines de l'IUT)
- Respect du cahier des charges minimal du programme : vous êtes allés au bout des niveaux niveaux 0 et 1, la mécanique du jeu est fonctionnelle.
- Spécifications et tests: chaque fonction doit documenter sa spécification, et les fonctions critiques doivent être accompagnées de leur fonction de test (test unitaire) dans un fichier séparé.
- La qualité du code : commentaires, découpage en fonctions, constantes, nommage correct…
- Considérer tous les cas particuliers d'un programme (ex: tel paramètre fait-il planter le programme?)
- Vous êtes allés plus loin en fournissant une interface graphique basique.
- Votre interface graphique est avancée, pratique, s'adapte en fonction du nombre de joueur.
- Incorporer des solutions techniques innovantes : soyez imaginatifs! Ici on note ce que vous apportez en plus sur le programme par rapport à ce qui est demandé. Attention : vous ne devez pas changer les règles du jeu.
- Vous avez implémenté des IA élémentaires pour pouvoir jouer sans adversaire.

Et maintenant ?

Go ! N'hésitez surtout pas à questionner vos enseignant.e.s si vous avez des questions. De plus, un salon dédié à la SAE permettra de dialoguer sur Discord. Bon travail ! Amusez-vous bien…