Rapport final du projet

BERT Audran, DALLON Damien, HUVIER Alexis

Lien du git
Langage : Python
Nous avons choisi le langage de programmation interprété Python car c'est le langage avec lequel nous sommes le plus à l'aise. De plus, ce dernier permet de faire de la programmation orienté objet tout en étant simple d'utilisation. Il est aussi largement utilisé pour réaliser des simulations.
Il a de plus permis via le module Pygame de créer une interface graphique assez complexe permettant une gestion complète de la simulation à partir de celle-ci.

Les explications liées au fonctionnement et au lancement du projet se trouvent dans le Readme du projet.

Pour les diagrammes voir pièces jointes au rapport.

Courte explication des diagrammes :

Diagramme de cas d'utilisation du joueur :
Le joueur peut donc créer une nouvelle partie, pour cela il doit choisir une classe. Il peut aussi charger une partie. Une fois en partie le joueur peut déplacer le personnage, mettre en pause pour pouvoir sauvegarder, reprendre ou quitter. Pour sauvegarder il doit préciser le nom de la sauvegarde.
Par rapport à la version précédente du diagramme, nous avons rajouté le cas d'utilisation "ControlerPersonnage" car nous avons finalement décidé que le joueur contrôlera lui-même le personnage.

Diagramme de cas d'utilisation du personnage :
Le personnage peut donc se déplacer, il peut arriver en se déplaçant qu'il se prenne un piège qui est lié au type de case sur lequel il se trouve. Pour se déplacer il doit choisir un mode de déplacement qui dépend de la case sur laquelle il se déplace ainsi que de sa classe (par exemple, le hippie ne peut pas prendre la voiture). Il existe la voiture, le vélo et les pieds. De plus, sur chaque action (se déplacer ou entrer dans un bâtiment) il peut tomber malade ce qui lui fera perdre de la vie. Si sa vie ou une autre de ses barres (life, hydration, mentality, satiety) venait à tomber à 0 le personnage meurt. Il peut donc aussi entrer dans un bâtiment ce qui aura un effet spécifique sur ses barres en fonction du bâtiment. L'université permet de gagner des diplômes qui sont l'objectif du jeu.

Diagramme d'états transitions :
Tout d'abord en lançant le jeu, le joueur arrive sur le menu où via des boutons il peut choisir de créer une nouvelle partie ou en charger. S' il choisit de créer une nouvelle partie il devra choisir le personnage et s' il décide de charger une sauvegarde il devra entrer le nom de celle-ci. Pendant le jeu, le joueur peut déplacer le personnage en appuyant sur les flèches, il peut aussi le mettre en pause. Cela permet de sauvegarder, de reprendre ou de quitter le jeu.
Par rapport à la version précédente du diagramme, nous avons rajouté la transition "Appui Boutton/ sedeplacer()" à l'état "Simulation en cours", toujours en rapport avec la décision du contrôle par le joueur.

Diagramme de classes :
La "Map" est composée d'une liste de listes de "Tile". Ces "Tile" peuvent être des "Building" ou bien des cases telles que un "SideWalk" ou une "Road" ou bien encore une "Forest" pour ne citer qu'elles.La classe "Building" hérite donc de "Tile", il y a 6 différents "Building" qui ont chacun leur propre effets. Les "Tile" ont

chacun un png qui est leur affichage en jeu. Elles ont aussi une fonction “can_go”, “apply” ainsi que “display”.
La classe “Character” regroupe les données d’un personnage qui se décline en trois types : “Standard”, “Hippy”, “HurriedMan”. La méthode “move()” permet au personnage de se déplacer sur la map en appliquant l’effet de l’action et celui du déplacement selon le mode.
La méthode “display(screen)” permet d’afficher le personnage sur la map.
Les classes “Movement” et “Trap” regroupent les méthodes permettant respectivement d’appliquer les effets du déplacement selon le mode et les effets des pièges.
Les classes “MainMenu”, “PauseMenu”, “SaveMenu” et “LoadMenu” représentent les quatres menus du jeu en utilisant une fonction “display” d’affichage et une fonction de gestion des évènements principalement.
La classe “Save” est utilisée directement par les deux derniers menus pour respectivement sauvegarder et charger la partie.
Pour finir, la classe UMLProjet relie toutes les classes en elle, en proposant des fonctions pour lancer ou arrêter la partie ou encore de changer d’écran affiché (Menu ou Jeu). C’est elle qui gère la boucle de jeu ainsi que la distribution des événements.
Le Main est externe au classe et se contente de parser les arguments console et de lancer la classe UMLProjet.
Par rapport à la version précédente du diagramme, nous avons ajouté quelques méthodes et pas mal d’attributs que nous n’avions pas prévu, notamment à cause de l’interface graphique ce qui a créé des différences au niveau du classe principale et des différents menus. Nous avons aussi détaché la classe “Save” de la classe principale car elle ne l'utilise pas directement.

Planning prévisionnel :

Planning réel du projet :

Légende du diagramme de GANTT :
Bleu pâle : activité commune
Bleu foncé : Audran Bert
Vert : Damien Dallon
Rouge : Alexis Huvier

Nous avons eu du mal à respecter le planning à cause des vacances et de la charge de travail des autres matières. Certaines étapes ont été réalisées de manière incrémentale, c’est à dire qu’une partie des

fonctionnalités ont été réalisées pour permettre d'avoir un projet fonctionnel puis les autres fonctionnalités ont été ajoutées au fur et à mesure. De plus, le logiciel ne permettait pas de montrer cette approche incrémentale. Certaines étapes comme la génération de la map ont été plus longues que prévu car nous n'étions pas satisfait de la génération.

Les différents choix réalisés :

Le choix le plus important que nous avons fait est que le personnage se déplace via les flèches du clavier et donc c'est le joueur qui contrôle et non une IA comme prévu au départ car trop complexe à réaliser.
Ensuite les bâtiments sont considérés comme route et trottoir en même temps, c'est-à-dire qu'on peut y accéder via n'importe quel moyen de locomotion.
Les trottoirs et forêts ne sont accessibles que si le personnage est à pied.
Les routes sont accessibles seulement en voiture ou vélo.
Tous les bâtiments sont reliés entre eux via au minimum une route ou un trottoir.
On a décidé de ne pas appliquer les effets du mode de déplacement (vélo, voiture, pied) lorsqu'on se déplace sur un bâtiment, cela a pour effet de prolonger un peu la simulation et de donner un intérêt aux bâtiments car sinon les bâtiments se contentent de régénérer ce qu'on vient de perdre.

Conclusion :

Toutes les fonctionnalités demandées ont été implémentées sauf l'ia qui a été remplacée par la gestion des contrôles (via les flèches) car nous avons eu des difficultés à implémenter un pathfinding fonctionnel et performant. Le choix du mode de déplacement est fait en fonction de la case sur laquelle il se déplace et s' il y a plusieurs modes de déplacement possibles il est choisi aléatoirement.

Nous avons réussi à atteindre 12 diplômes lors d'une partie sur une carte de 5*5 avec l'homme standard.

Nous avons réussi à rester assez organisé et à travailler assez régulièrement sur le projet ce qui fait que nous n'avons pas eu de difficulté à finir dans les temps. Le projet était intéressant et la difficulté bien dosée, cela a donc été une bonne expérience pour nous trois et nous a permis d'apprendre à bien structurer un projet.