Projet de Programmation Impérative

Bazar Bizarre (ou Bazaris : The bizarre adventure)

Principe du jeu de base :

Il s'agit d'un jeu de cartes avec un certain nombre de pion. Les cartes contiennent un nombre de dessin (personnages) adapté au nombre de pion. Ces dessins peuvent être le dessin exact d'un pion, dans ce cas, il faut attraper le pion en question. Si aucun pion n'est dessiné sur la carte, il faut attraper le pion qui n'a aucune caractéristique commune avec les dessins de la carte. Dans le cadre de ce projet, vous jouerez contre notre intelligence artificielle, Josephin. Le thème de notre jeu est un combat de jeu RPG (univers fantaisiste).

Options pratiques :

Vous avez tout d'abord un menu d'options vous permettant de choisir une expérience de jeu plus adapté à votre niveau. Vous pouvez choisir entre trois difficultés (facile, moyen et difficile), celles-ci vont influencer sur le temps de réaction et le nombre de points de vie de Josephin, notre intelligence artificielle et votre ennemi. Vous avez aussi le choix de jouer avec 5 ou 6 pions. Dans le cas où il y a 5 pions il n'y aura que 2 personnages sur la carte tandis que dans le cas où vous jouez avec 6 pions il y en aura 3, cela rajoute un petit peu de défi au jeu. Je vous recommande de jouer en difficile, 6 pions, rien de mieux.

Un choix de pion est l'équivalent d'une attaque. Vous avez des combattant en face de vous possédant chacun une classe et une couleur à eux. La carte sera matérialisée non pas comme une carte mais comme une équipe, celle des sbires de Josephin. Un seul de vos combattant est apte à combattre l'équipe adverse donc il est recommandé d'envoyer le bon compagnon.

Vous possédez une barre de vie et il en va de même pour Josephin.

Dans le cas où vous prenez trop de temps pour attaquer, Josephin prendra cette occasion et vous attaquera. Les dégâts qu'il vous inflige dépondront du temps qu'il a mis pour vous attaquer, si vous réussissez à l'attaquer avant il en va de même pour vous. Mais bien sur personne n'est parfait, Josephin peut rater son attaque, il s'infligera alors 150 dégâts. Si vous envoyer le mauvais combattant au front, il est sûr qu'il ne mènera pas à bien sa mission et par sa maladresse vous infligera 150 dégâts également.

A chacune des attaque, alliée ou ennemie, une suite de texte sera générée dans un cadre en bas à droite de votre fenêtre pour donner un aspect « jeu de rôle ». Le texte généré va varier en fonction de la situation. Attaque alliée ou ennemie, réussite ou échec, l'attaquant choisi pour attaquer. Il y a différentes phrases pour les différentes situations qui seront choisies aléatoirement pour éviter la monotonie.

Lorsque la vie de Josephin tombe sous certains seuils il subit des transformations accompagnées d'améliorations. A chaque transformation, son temps de réaction est amélioré et son aspect physique en jette de plus en plus. En mode difficile à la dernière phase son temps de réaction est incroyable.

Comme dans la plupart des jeux, il vous suffit de faire tomber les points de vie du boss à 0 pour obtenir la victoire mais si vous parvenez à vous faire dominer par Josephin et que celui-ci fait tomber votre santé à son plus bas ce sera évidemment le Game Over pour vous. A la fin d'une partie vous avez le choix de soit recommencer une partie et de redéfinir les paramètres de départ ou bien tout simplement de quitter.

Vous l'aurez compris, votre unique but va être de défaire les plans de Josephin sous sa meilleure forme, il faudra vaincre Josephin en mode Difficile, 6 pions pour pouvoir se considérer comme un vainqueur.

Structure du code :

- Le code concernant le jeu en lui-même (carte, vie, temps, appel des fonctions) dans le module Bazaris.
- Le code concernant le décor du jeu, pratiquement toute la mise en place du décor dans le module Décor.
- Le code concernant la génération de texte, le scénario de chaque situation dans le module Texte.
- Le code concernant le dessin des différents personnages dans un module « personnages ».

A savoir que pour la création de chacun de ces modules nous avons utilisé des modules de base (time, random et turtle) ainsi qu'un module polygone construit afin de faciliter notre développement du code, celui-ci contient de nombreuses fonctions pratiques permettant d'économiser des lignes et de ne pas trop s'embêter.

Les personnages :

Il y a donc 6 personnages (pions), chacun dans un module contenant sa fonction d'appel (ainsi des fonctions de dessin des parties du corps parfois), ainsi qu'un boss dans le module JosephLeBoss contenant sa fonction d'appel, les fonctions de dessin de ses améliorations et d'autres fonctions (génération cactus dans le décor, barre de vie) car elle partage certaines mécaniques. Les 6 fonctions d'appels des pions ont les mêmes paramètres : coordonnées, couleur et facteur de taille pour les adapter à une même échelle et changer la taille des sbires.

A savoir que la mage et le pistolero ne sont pas fait de formes élémentaires car ce sont les premiers réalisés et je ne connaissais pas la consigne mais le reste n'est que formes.

On peut sauter et accélérer la génération du texte avec la barre espace (capture de l'appui de touche).

Le texte :

Pour le texte généré nous avons choisi une approche particulière permettant de donner un aspect plus jeu. Le texte se génère de façon progressive et non instantané pour correspondre à la plupart des jeux du marché. Pour cela nous décortiquons simplement le texte caractère par caractère et marquons un temps de pause entre la génération de chaque caractère. Nous utilisons la fonction write() de turtle pour afficher notre texte.

Le texte est organisé sous une forme de scénario en quelque sorte car les phrases dépendent de la situation et les phrases peuvent différer selon le hasard (3 possibilités de phrases pour certaines situations).

Le décor :

Le module de décor contient les différentes fonctions de dessin des éléments fixes (sauf les pions).

Le décor est entièrement fait de formes élémentaires colorées et de lignes solitaires. Toutes les fonctions des formes ont été écrites dans le module polygone.

Le déroulement du jeu, Bazaris :

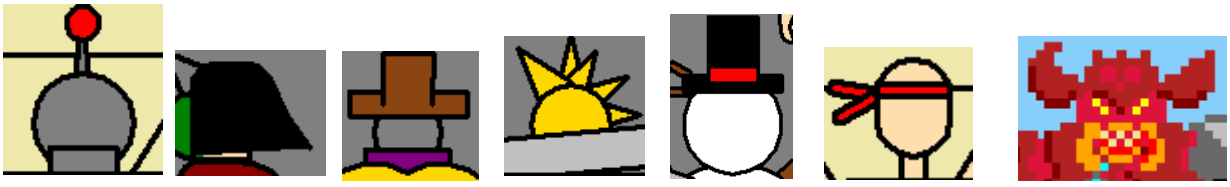
Bazaris est le programme qui appelle tout le reste car toute l'organisation du jeu se fait dans ce fichier. Ce fichier contient des fonctions comme celles de la génération de la carte et donc de l'équipe adverse, celle des « hitbox » soit la zone de clique de chaque pions, le menu de départ ainsi que celui de fin, et la fonction du déroulement du jeu.

La fonction jeu() prend en compte les choix fait dans le menu, et elle est constituée de nombreuses boucles dont la principale qui ne s'arrêtent pas tant que les points de vie du joueur ou du boss ne sont pas tomber à 0. On utilise des fonctions comme onscreenclick(choix) pour capturer le choix du joueur, time() et tracer(0,delay=_) pour calculer le temps de réaction du joueur, ce n'est pas une manière très précise, nous avons essayé d'utiliser la fonction ontimer() mais elle nous a posé quelques problème nous avons utilisé une turtle comme une sorte de sablier.

Cette fonction appelle aussi la fonction des barres de vie et celles des points de vie perdu, ainsi que les fonctions de scénario, de générations de textes.

La fonction jeu() est donc en quelque sorte le noyau du jeu dans lequel se concentre la plupart des informations.

Pions et boss :



Robot Mage Pistolero Guerrier Homme-neige Moine Josephin(boss)

Problèmes rencontrés et solutions :

Le premier problème rencontrée était le fait de mettre un temps de réaction réaliste à l'IA, nous ne pouvions pas utiliser de fonction comme sleep() car celle-ci mettent en pause tout le programme nous empêchant de répondre dans ces intervalles. Nous avons donc décidé de nous servir d'une turtle qui

avance avec la fonction tracer() et l'argument delay, à chaque update nous prenons le temps écoulé depuis le dernier update() et l'ajoutons au temps écoulé total afin de déterminer le temps de réaction.

Le deuxième problème concernait la génération des sbires du boss. Comme vous avez surement pu le voir la carte n'est pas vraiment une carte, nous ne pouvons pas uniquement dessiner un rectangle par-dessus les personnages pour en dessiner un autre car ceux-ci sont fondus au décor au reste, dessiner un rectangle par-dessus serait ruiner l'aspect qu'on voulait donner au jeu. Nous pensions d'abord définir une turtle pour chaque sbire afin de pouvoir reset() facilement mais cela reviendrait à changer tous les fonctions de dessin de personnage, cela été possible et ça nous a d'ailleurs fait découvrir des choses mais nous avons fini par trouver une solution bien plus simple, une sorte de sauvegarde et suivie d'un chargement. On se déplace à des coordonnées sur lesquelles le jeu ne va pas repasser (on considère cela comme un point de sauvegarde) puis lorsqu'on veut effacer nos personnages on annule toutes les actions faites jusqu'à avoir attient les coordonnées du point de sauvegarde avec une boucle et la fonction undo(). Ainsi on peut dessiner les nouveaux personnages sur l'espace vide crée. On utilise aussi cette technique pour effacer la phrase générée dans le cadre du texte afin d'en générer une nouvelle, certes on aurait juste pu dessiner un rectangle noir par-dessus mais c'est mieux.

Un des plus gros défis auquel nous avons fait face était d'employer le moins de ressources globales dans le programme pour éviter certains problèmes, comme le fait d'utiliser une turtle et ses coordonnées pour capturer le click à la place de variable modulable.

Et sinon le problème qui est toujours d'actualités mais qui semble assez normal est le fait que le jeu malgré le fait qu'il s'adapte en partie à la résolution des différents ordinateurs, peut toujours générer des éléments du jeu à des endroits où il ne devrait pas être…

Déception majeure :

Le jeu aurait été bien plus beau avec certaines animations de personnages, d'effets de dégâts et autres, mais la façon dont laquelle le code a été écrit aux premiers abords ne nous a pas permis de modifier librement et sans conséquences notre code. Donc pour permettre des animations il aurait fallu réécrire la majeure partie des fonctions ce qui a instantanément provoqué une certaine flemme en nous mais si un jour en dehors du cadre du projet nous trouvons la foi de reprendre ce projet afin de l'améliorer pour le seul accomplissement personnel, nous réécrirons le code jusqu'à ce qu'ils correspondent parfaitement à l'idée que l'on s'était fait de lui, notre perfection.

Ah oui et aussi le fait qu'on ne puisse pas superposer deux sons avec winsound donc on ne peut pas mettre de bruits d'attaque par-dessus la musique ce qui me donne un blues extrême.

Répartition du travail :

Achraf :

- Dessin de tous les pions.
- Dessin du décor (sauf pour les cactus).
- Génération et affichage du texte en fonction des situations.
- Les hitbox (zones valides de click) (calcul des coordonnées de la zone en fonction de la taille, calcul sur papier, et annulation du click hors des zones)
- Création du module de formes polygone.
- Génération des personnages en fonction de la carte.
- Harmonisation du jeu, mise en place des éléments aux bons endroits pour donner un bon aspect.
- Minuteur permettant la réalisation du temps d'attente du boss.
- Ecriture du rapport et choix du thème du jeu.
- Interaction au clavier pour accélérer et passer le texte généré.
- Musique et effets sonores

Enzo :

- Génération de la carte (plus dur de tout le programme).
- Dessin du boss en pixel et création du boss (temps de réaction, taux de réussite, phases du boss incluant les améliorations graphiques et de ses compétences).
- Dessins des cactus et génération des cactus.
- Barre de vie du joueur et du boss, ainsi que le système de perte de vie.
- Menu de début et écran de fin (aspect et boutons).
- Les hitbox (système de click avec une turtle après choix du pion).
- Scores adaptatifs, selon la phase du boss.
- Annulation des dessins avec la technique du « point de sauvegarde ».
- Affichage des règles.

Travaux fait à deux :

- Beaucoup de choses à vrai dire.
- L'un peut corriger les bugs de l'autre.
- Mais l'un peut aussi simplifier les programmes de l'autre.
- Donc nous avons en quelque sorte majoritairement fait chaque partie du projet à deux.

Impressions sur le projet :

Ce projet était très intéressant car il nous a permis de découvrir de multiples choses qui nous serviront à l'avenir. Lors de sa réalisation nous avons toujours essayé de dépasser nos limites.

A chaque problème existe une solution et le plus amusant dans la réalisation de ce projet était sûrement la phase de recherche, délicate et intense, qui nous menait souvent dans des impasses à laquelle une nouvelle phase de recherche encore plus intense était menée.

Avec toutes les informations accumulées nous avons essayé d'offrir à notre jeu exactement ce qu'on voulait de lui, on aurait évidemment pu ajouter bien des fonctionnalités, nous en avions plusieurs en têtes mais nous ne voulions pas trop saturer le jeu.

Après avoir trouvé toutes les solutions à nos problèmes nous avons aussi pu apporter du soutien à certains camarades de classes, ce qui apportait en même temps une sensation d'accomplissement et de puissance intellectuelle.

Avoir choisi le contrat +++++++ était sûrement le meilleur choix que nous aurions pu faire, toujours choisir la plus haute difficulté afin d'en apprendre un maximum.

EL BACH Achraf MANUEL Enzo

Polytech Nice Sophia , Peip1 Groupe 5