PROJET C# - M1 EIF

Ialgen ALLAL & Bertrand GOULOUMES

1. Note explicative sur les règles du jeu :

Notre jeu est inspiré du jeu *Space Invaders*. Ce dernier est un jeu d'arcade en 2 dimensions. Nous avons gardé la mécanique de base où le joueur représenté par un vaisseau doit faire face à des vagues ennemies et peut leur tirer dessus en tapant sur la Barre d'espace et se déplacer latéralement/horizontalement (flèches de gauche et de droite) dans les limites de la Box (cadre blanc).

Nous avons ajouté des mécaniques différentes et un thème. En effet le jeu s'appelle **Space Rebellion**, faisant allusion à la saga Star Wars (par le design de la page d'accueil, l'allure des vaisseaux ennemis : chasseurs de l'Empire).

Les variantes sont que les chasseurs apparaissent tous par le coin nord-ouest (ou haut gauche) de la box a rythme dépendant du niveau de jeu choisit par le joueur. Et se déplacent horizontalement jusqu'à rencontrer un mur et descendre d'une ligne et de partir dans le sens inverse.

Le jeu se finit lorsqu'un chasseur de l'Empire (appelé *Invader* dans le code) est sur le point de se retrouver sur la même ligne que le vaisseau de la Résistance (appelé *DefenderShip* dans le code), le rendant vulnérable et annonçant la fin de la Résistance.

Vous pouvez augmenter la difficulté selon le pilote sélectionné ce qui accélèrera la vitesse des chasseurs ennemis.

Vous ne pouvez plus tirer de missiles tant qu'il en reste une dans la Box ou que ce missile n'a pas touché un ennemi.

Il est évidemment possible de rejouer, votre score (nombre de vaisseaux abattus et temps de survie) sera affiché.

2. Note explicative sur le fonctionnement du code :

Le jeu est composé de 3 fichiers de code principaux : *Program.cs*, *HomePage.cs* et *Game.cs*. Puis de 4 fichiers contenant les codes de 4 classes d'objets du jeu : *DefenderShip.cs* (vaisseau contrôlé par le joueur), *Invader.cs* (vaisseau ennemi), *Box.cs* (cadre de jeu et aussi de la page d'accueil), *Bullet.cs* (missiles envoyés par les vaisseaux).

La classe *HomePage* gère toute ce qui se passe avant et après une manche de jeu. La classe *Game* gère une manche de jeu unique.

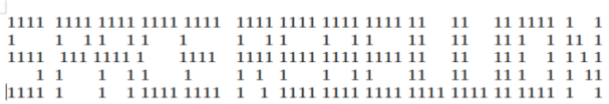
HomePage.cs

Les 2 méthodes importantes sont *Launch* et *Draw*.

Launch gère le dérouler des appels de *Draw* et récupère les deux réponses du joueur et on les renvoie pou paramétrer la création de la classe *Game*. La méthode gère aussi si le joueur ne veut plus rejouer.

Draw fait appel à *Random_Invaders* et *Random_Bullets* pour afficher des vaisseaux et des tirs sur la page d'accueil. On se sert au fur et à mesure de *working_on_row* pour déterminer la ligne sur laquelle on affiche.

Puis *Game_Title* permet d'afficher le titre. En effet, on ne pas peut augmenter la taille des caractères dans la console donc l'alternative à été de dessiner le texte sur un fichier Word avec des 1 comme on peut le voir sur l'impression écran suivante :



Il suffit ensuite d'automatiquement remplacer les vides par des 0, et *Game_Title* peut écrire le titre dans la console. Enfin grâce à 2 booléens (*restart* et *phase_1*) le code sait si on a déjà fait une partie et si déjà eu le premier message (demandant de jouer ou rejouer : *Phase_1* ou *EndGame_Announcement*).

Game.cs

La classe *Game* contient des méthodes commençant par *Update* utilisé dans la gestion des positions des éléments du jeu. A chaque rafraichissement de l'affichage, les méthodes updates sont appelées pour définir les nouvelles positions des vaisseaux et missiles présents dans l'espace de jeu.

Une méthode *BulletInvaderCollisions* est appelé lors de l'actualisation des positions pour détecte d'éventuelles collisions entre un missile et un vaisseau. La collision entrain la destruction du vaisseau et donc le retrait de l'instance de classe qui le définit.

Une méthode *EndGame* permet de déterminer à chaque actualisation si un vaisseau ennemi est arrivé sur la ligne du vaisseau du joueur, si c'est le cas, la méthode renvoie le booléen *false* et la boucle du jeu ne se répète pas.

Une méthode *InitPath* permet de déterminer le chemin de position que les vaisseaux ennemis parcourent selon la dimension de l'espace de jeu définit indépendamment.

Program.cs

La classe *Program* correspond au déroulé du jeu, c'est le *Main* de notre code. Il fait donc appel à *HomePage* pour gérer l'affichage d'avant partie et récupérer les requêtes du joueur (s'il veut jouer et si oui avec quelle difficulté).

Les 2 boucles *while* imbriquées sont très importantes à comprendre. On reste dans la première tant que le joueur n'a pas dit qu'il ne voulait plus jouer (il peut le dire avant et après une manche en cliquant sur une touche différente de la *SPACEBAR*). On est dans la seconde tant que les vaisseaux ennemis n'ont pas gagné.

Dans cette seconde boucle, le programme commence par effacer l'affichage de la console. Puis on récupère la touche sur laquelle le joueur a cliqué (si il clique sur une touche), on met à jour la position des éléments mobiles (*Bullet*, *Defendership* et *Invader*) et on réaffiche tous les éléments mobiles et non mobile composant le jeu. Cette boucle n'est pas réintégrée au passage suivant si un vaisseau de type *Invader* est arrivé sur la ligne de position du *DefenderShip*. Le booléen renvoyé par la méthode *EndGame* est alors *true*.