Rapport de projet :
Programmation Orientée Objet C#
"Space Invaders"

Brackenier Jordan

10/12/2017

I - Sujet

Introduction

le but de ce projet est de reproduire un ancien Jeu vidéos, le "Space Invaders" en langage C#, donc en programmation orientée objet. Ce langage se basant sur le principe de créer des objets qui interagissent entre eux afin de réaliser l'application. Pour pouvoir créer l'application il est donc nécessaire de pouvoir accéder à des images en stock par exemple, que l'on appel des Ressources, ces dernières permettant de représenter les éléments du Jeu, ou encore des fichier texte par exemple pour ajouter des niveaux de difficulté à notre Jeu. De plus, une gestion du temps, et une interface graphique sont également requise, les objets matérialiser par les images, placer dans l'interface graphique évoluant en fonction du temps.
Nous avons développer ce projet sous Visual-Studio.

Dans un space Invaders, on contrôle donc un vaisseau se déplacent horizontalement, équipé d'un canon qui permet de détruire les ennemis qui se rapproche de vous.

Eléments fournies

Le squelette d'une application Balle qui tombe. Contenant une Classe Game permettant de géré la création des objets dans l'interface graphique, cette dernière étant générer dans les fichier Form1. Une classe Ballequitombe également présente m'a permit de débuter ma classe Spaceship.

II - Structure du programme

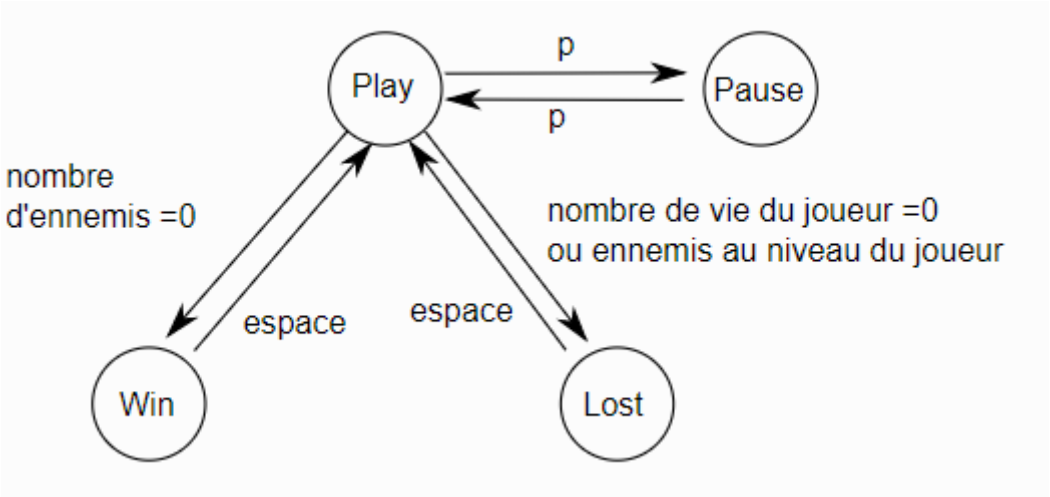
Vecteur2D

Vecteur2D simplifie les déplacement et les requêtes de position en 2 dimensions.

Game

Game initialise notre Jeu, la classe possède tous les objets du jeu dans une liste instancier à l'initialisation. Si ces objets ne sont plus considérer comme étant en vie, ils sont alors détruit. De plus, les différents états du Jeu y sont gérer en fonction de la situation, ces états sont afficher dans le jeu pas la méthode Draw. Ces états sont une énumération, Play, Pause, Win si le jeu est gagné, Lost si le jeu est perdu.

Le jeu est perdu si le nombre de vie du joueur est à zéros ou si les ennemies sont arrivés au niveau du joueur.



De plus, l'interface graphique y est générer en singleton, il s'agit d'une seule instanciation au plus de la classe Game.

GameObject

GameObject est la classe abstraite représentant tous les objets qui composent le jeu tout en définissant des méthodes abstraites communes, ainsi qu'une énumération de camp possible pour chaque objet. Soit Enemy pour les vaisseau ennemie, soit Ally pour le vaisseau du joueur ou Neutral pour les bunkers.

La méthode Collision pour la collision entre deux objets du jeu de camp différents. Draw pour dessiner chaque objets, IsAlive pour déterminer si les objets sont en vie et donc si ils doivent être supprimer du jeu dans la classe Game. Ainsi que la méthode Update qui fait évoluer les objets dans le temps, c'est à dire tout les deltaT.

Chaque nouvelle objet aura donc une classe qui lui est propre et héritera de GameObject.

SimpleObject

Classe abstraite héritant de GameObject, elle définit le comportement commun des classes Objets qui en hériterons en implémentant les méthodes de GameObject, une nouvelle méthode abstraite fait cependant sont apparition, il s'agit de OnCollision qui gérera le comportement des collision en fonction de la nature de l'objet toucher par un missile.

EnnemyBlock

Hérite de GameObject également, cette classe définit l'ensemble des vaisseau ennemies contenu dans une HashSet, c'est un Block (enemyships) dimensionner par une taille (Size), cette taille est redimensionner en permanence par la méthode UpdateSize appeler dans Update. Une méthode AddLine nous permet d'ajouter des lignes de vaisseau ennemies dans notre block. Une méthode EnemyShoot est également appeler dans Update pour tirer des missile aléatoirement à partir des vaisseau ennemies.

SpaceShip

Hérite de SimpleObject, cette classe représente l'ensemble des vaisseaux du jeux, alliés ou ennemis.

Missile

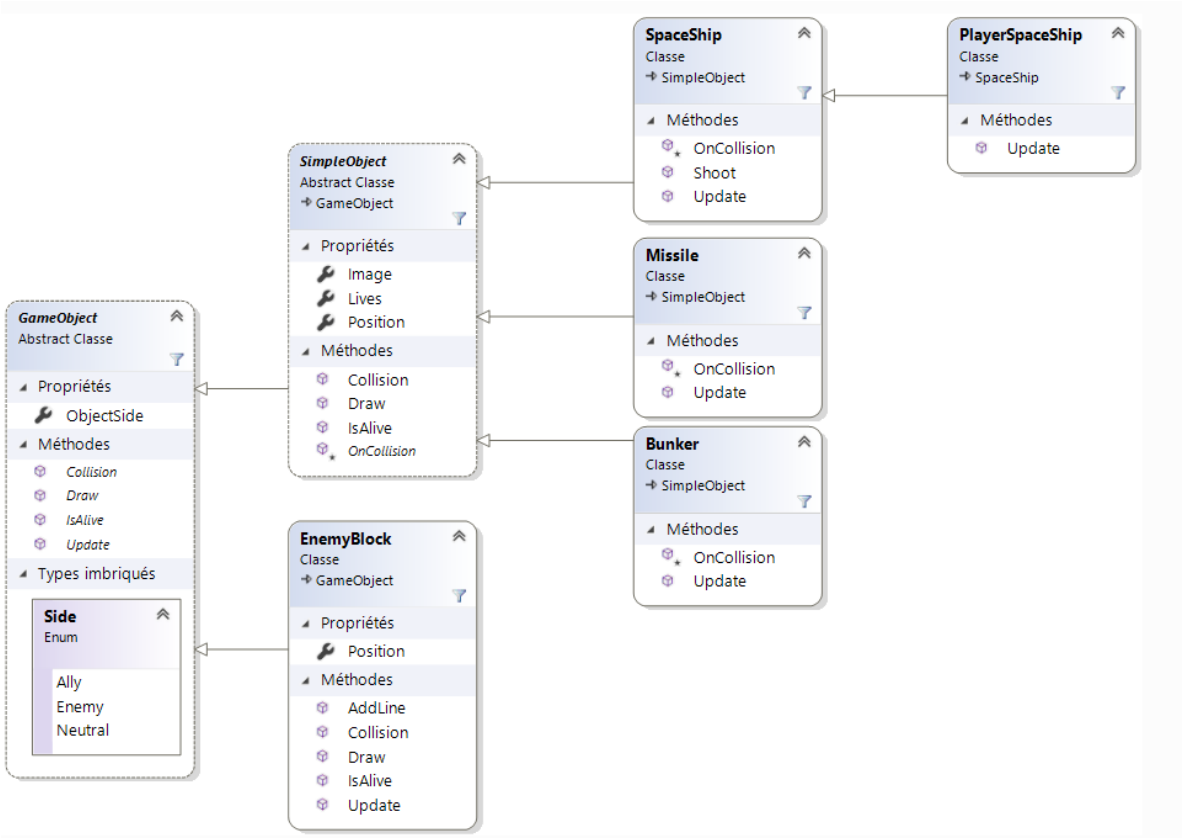
Hérite de SimpleObject, cette classe représente les différents missiles tirés par les vaisseaux du jeu.

Bunker

Hérite de SimpleObject, cette classe représente les différents bunkers.

PlayerSpaceShip

Hérite de SpaceShip, cette classe représente spécifiquement le vaisseau du joueur. Sa méthode Draw permet d'afficher les points de vie du joueurs en bas à gauche de l'interface.



J'ai respecter ce plan dans d'organisation des classes et méthodes dans mon programme, en y ajoutant la fonction EnemyShoot.

III - Problèmes rencontrés

Au début du projet, j'ai perdu beaucoup de temps en compréhension, un temps qui m'aurait été précieux afin de finaliser mon jeu et d'y ajouter les addons demander. Cependant d'autre tâche mon poser problème.

Notamment, la méthode UpdateSize et Update de la classe enemyblock, leurs algorithme ont été fastidieux à faire fonctionner correctement, il m'a fallut les tester et retester. J'ai fait afficher le rectangle autour de l'enemyblock pour le voir évoluer, il évoluer souvent sans collision et juste en fonction du déplacement de l'enemyblock, j'ai cependant petit à petit corriger ces bugs.

D'autres algorithme m'ont également poser problème, mais par ma persévérance et l'aide que j'ai pu recevoir en cours, j'en suis venu à bout !

Conclusion

Le projet est très intéressant et donne envie d'en apprendre plus sur le développement orienté objet afin de pouvoir soit même réaliser des projets par la suite, cela m'a permit de découvrir un nouveau langage, mais également d'améliorer ma façon de penser pour mettre en place des algorithme, afin de programmer efficacement en temps et qualité. Cependant, je pense encore avoir beaucoup de progrès à faire affin d'évoluer par moi-même.